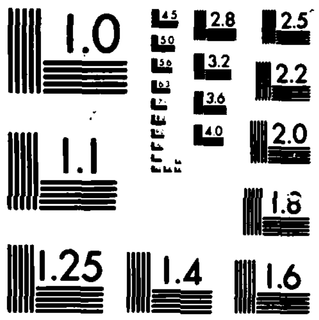
AD-A164 100 DESIGN OF THE HUMAN-COMPUTER INTERFACE FOR A COMPUTER AIDED DESIGN TOOL F..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. T C MALLARY 1/2

UNCLASSIFIED DEC 85 AFIT/GCS/ENG/85D-8 F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

1

DESIGN OF THE HUMAN-COMPUTER INTERFACE FOR A COMPUTER AIDED DESIGN TOOL FOR THE NORMALIZATION OF RELATIONS

THESIS

Thomas C. Mallary
Captain, USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

(1)

AFIT/GCS/ENG/85D-8

DESIGN OF THE HUMAN-COMPUTER INTERFACE FOR A COMPUTER AIDED DESIGN TOOL FOR THE NORMALIZATION OF RELATIONS

THESIS

Thomas C. Mallary
Captain, USAF

AFIT/GCS/ENG/85D-8

AFIT/GCS/ENG/85D-8

DESIGN OF THE HUMAN-COMPUTER INTERFACE FOR A COMPUTER AIDED DESIGN TOOL FOR THE NORMALIZATION OF RELATIONS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science

Thomas C. Mallary, B.S.
Captain, USAF

December 1985

Preface

The term "user friendly computer interface" is one that has intrigued me during my eight years of experience in operational flying units. During that time period, I have had the opportunity to observe the frustration that people, varying in rank from Airman to Colonel, can experience when their supposed user-friendly computer system suddenly becomes hostile!

These observations, together with my own experiences with less than friendly systems, provided the motivation to get involved in a project that dealt with the human-computer interface issue. The thesis effort has given me the opportunity to investigate a number of design issues and techniques, and has also revealed that the design of a "friendly" system is not a trivial task.

I would like to extend my thanks to Dr. Thomas Hartrum for the help he provided in his role as thesis advisor. Thanks also go to Capt Steve Woffinden and Dr. Henry Potoczny, who as committee members, gave me valuable feedback used to complete the project, and to Professor Dan Reynolds who provided advice and assistance on the statistical analysis performed during the effort. Finally a word of thanks to Dan Zambon and fellow GCS-85D students, who could always be counted on to help with the frequent system and syntax problems that managed to surface during the past eight months.

Table of Contents

Page

* This appendix is contained in Volume II of the Thesis. This material is maintained by the Department of Electrical Engineering, Air Force Institute of Technology.

List of Figures

AFIT/GE/ENG/82D-8

Abstract

The project involved the design and implementation of the human-computer interface of a computer aided design ~~(CAD)~~ tool used in conjunction with a relational database. The tool gives a database administrator an interactive means of specifying functional dependencies for a given relation. It then executes a nonloss decomposition, normalizing the original relation to third normal form.

Background information is provided on the fundamentals of a relational database and on the concept of functional dependencies. An understanding of the term "user-friendly system" is developed, together with a summary of state-of-the-art techniques and guidelines meant to help the system designer create more friendly interfaces.

The program development process used in the project is described in detail. Major design issues of the project are described, with emphasis placed on areas impacting the system's human-computer interface.

A tool is developed to measure and quantify the extent of user satisfaction in the software system. The measurement tool is implemented by a test group asked to evaluate the normalization tool interface. Results of the evaluation are presented.

DESIGN OF THE HUMAN COMPUTER INTERFACE FOR A COMPUTER AIDED DESIGN TOOL FOR THE NORMALIZATION OF RELATIONS

I. Introduction

Background

A computer database is a record keeping system whose purpose is to accurately record and maintain information (7:3). A database administrator (DBA) is responsible for the design, development and maintenance of such a system and has overall control of the stored information. Some of the DBA's design tasks are to define the contents of the database and to decide the schema or structure of the system. These tasks can be very complex and time consuming, particularly for a large database management system (DBMS).

It may be possible to help the DBA perform some of these complex design tasks by developing computer aided design (CAD) tools. CAD tools, already used in computer hardware design applications, could aid in the performance of database design tasks that must now be done by hand. One such task is the normalization of relations, a process of establishing design constraints for a relational database system. In the normalization process, the DBA must define dependencies between the attributes of a database and combine the attributes into relations or tables. The application of normal forms (constraints) can help reduce data redundancies and leads to a high level of data reliability. Reduction of redundancy and maximization of data reliability or integrity are necessary objectives of any DBMS.

There are several factors that make the normalization process a reasonable candidate for the development of a design tool; the process is time consuming and prone to human error, as well as being fairly well defined and adaptable to automation. Two recent AFIT thesis efforts dealt with the topic of CAD tools for the normalization of relations. Travis (31) developed a program to normalize relations in third normal form (3NF). This program was enhanced by Jankus (16) in 1984, by developing an interactive normalization software tool for use on the LSI-11 computer. This tool consisted of two main parts: an interactive package that enabled the user to specify functional dependencies from a given list of attributes, and a non-interactive portion that performed the actual normalization of relations once the functional dependencies were determined.

The interactive portion of the CAD tool used two CRT screens, one to display information and instructions to the user, the other to display a list of attributes from which the user determined functional dependencies. The tool provided the user with two selection techniques, using either the keyboard or a mouse, to select the determinant and dependent attributes for a given relation. The user could continue the selection process until all desired relations were specified.

Problem

The purpose of this thesis effort was to enhance Jankus' CAD normalization tool with respect to both computer input and output. In terms of input, the problem was to improve or redesign the interactive portion of the tool. To solve this problem, it was first necessary to examine the manual process which a database administrator goes through in determining functional dependencies. Having identified this process, it was then necessary to determine how a computer interface could be designed to better aid the DBA in accomplishing this task.

Existing program output was also evaluated and software changes made where necessary, to provide a more readable data output format for the system user. Under the existing system, the user may print out a copy of data input and output files. These files are in a format that can be easily read by the system, but the format is difficult for the user to read.

Scope

The scope of this effort was limited to the enhancement of the Travis/Jankus CAD tool. No attempt was made to develop alternative algorithms to accomplish normalization in 3NF. Rather the thesis effort concentrated on the improvement of the front and back ends (input and output) of the existing system. Emphasis was placed on researching state of the art techniques in human computer interface, and incorporating human factor concepts in the design of a more user-friendly system.

Approach

The approach taken in this study was to first evaluate the existing system in terms of user-friendliness. In early 1985, Jankus' CAD tool had been used by a group of AFIT EENG 646 students for a computer data base system class project. The critiques provided by these students were reviewed for feedback on the features of the tool which were helpful, and to determine those aspects of the tool which were either confusing or "unfriendly" from a user's point of view. With the student critiques in mind, a literature search was then conducted to gain a better understanding of current techniques used in the design of user friendly systems. Using the student feedback and insight gained from the literature review, modifications were then made to enhance Jankus' software, using the interface devices currently available in the AFIT Digital Engineering Laboratory. Finally, the modified CAD Normalization Tool was evaluated by a group of data base students, familiar with the normalization process, to provide feedback on the system modifications.

Sequence of Presentation

In line with the general approach outlined above, this report consists of six sections. Chapter II gives a brief description of the interface techniques used in Jankus' CAD tool and a summary of the comments made by students who evaluated the tool. Having an understanding of some of the possible problem areas of the existing tool, Chapter III discusses the factors that must be considered in the design of a user friendly system, based on information found in current literature. Included in this section are a discussion of what is meant by a user-friendly interface, a presentation of design principles that should be used in creating the interface, and finally a discussion of a possible development process that can be used in the design of a user friendly program.

Using the design guidelines and process presented in Chapter III, Chapter IV describes the system requirements and design process used in developing the software for this thesis effort. Chapter V expands on some of the project's more critical design features. Several major program modules will be discussed, giving a better insight into how the normalization tool is used. Chapter VI discusses the results of student evaluations of both the modified and unmodified versions of the CAD tool. Finally, Chapter VII presents the conclusions and recommendations for the thesis effort.

II. System Background

In order to provide a better understanding of the function of the CAD normalization tool developed in this project, this chapter presents a more detailed view of the tool developed by Jankus. The purpose of this information is to give the reader a better insight into how the normalization tool fits into the over-all data base management system, to describe the user-computer interface of the Jankus tool, and to summarize the comments of students who used the normalization tool for a data base class project. This background information, particularly the discussion of student critiques of the user-friendliness of the Jankus tool, provides a point of reference for the remainder of this thesis effort.

System Environment.

The CAD normalization tool is only one element of the relational data base system being developed in the AFIT Digital Engineering Laboratory. Within the current configuration, the normalization tool is an optional program that the DBA may use in the design of a relational database. New relations and domain definitions are first defined using Kearns' (17) Data Base Manager (DBMGR) Program. DBMGR creates a file containing relations and domain definitions that may then be used as input to the normalization tool.

In using the normalization routine, the DBA is first given the opportunity to interactively specify functional dependencies that exist between the attributes of a given relation. Functional dependencies are special forms of integrity constraints that exist within a given relation. These dependencies are a function of the real-world relationships that exist between the various attributes of the relation. Ullman discusses the issue of functional dependencies when he says,

> The only way to determine the functional dependencies that hold for relation scheme R is to consider carefully what the attributes mean. In this sense, dependencies are actually assertions about the real world; they cannot be proven, but we might expect them to be enforced by a DBMS if told to do so by the database designer (32:215).

Date offers the following definition of functional dependence:

> Given a relation R, containing attributes X and Y, attribute Y is functionally dependent on attribute X if and only if each X value has associated with it precisely one Y value (7:240).

For example, relation First shown in Figure 1 contains the attributes S#, P#, status, city, and quantity. In this supplier, parts relation, S# determines the supplier's status and city in which the supplier is located, the S# and P# together determine the quantity on hand, and the city in which the supplier is located determines the supplier's status. These constraints or functional dependencies could be represented in the form of a functional dependency diagram as shown in Figure 2. By specifying the functional dependencies and displaying them in diagram form, it is possible to identify certain problems with the relation that can lead to database update anomalies. By examining the functional

S#	P#	Status	City	Quantity
S1	P1	20	London	300
S1	P2	20	London	200
S2	P3	10	New York	150
S2	P4	10	New York	100
S3	P5	30	Paris	300

Figure 1. Sample Relation "First"

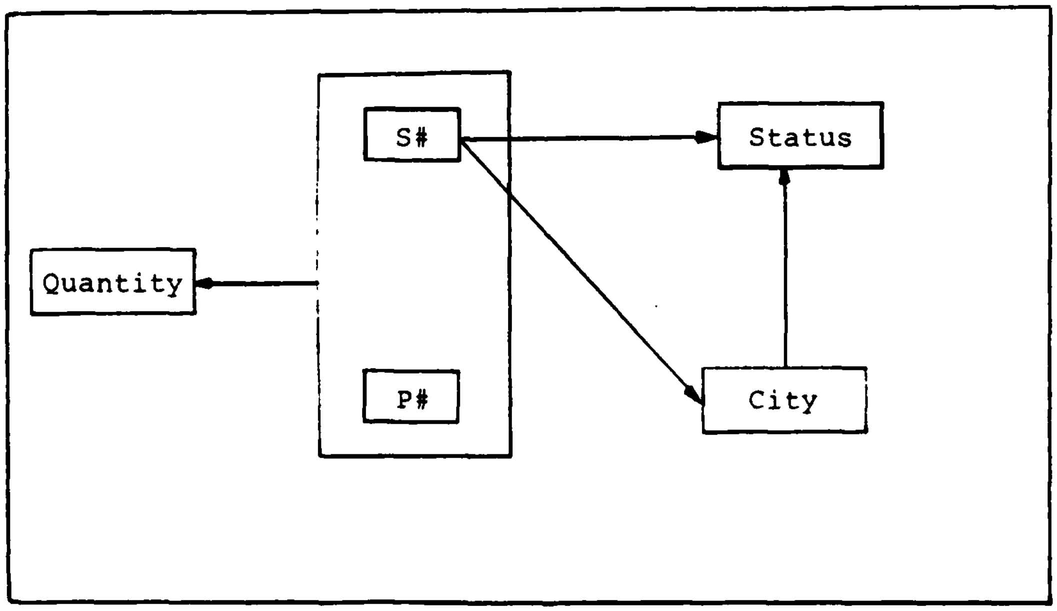


Figure 2. Functional Dependency Diagram (7:243)

dependencies, the DBA can recognize unwanted dependencies among the relation attributes, and break the relation down into smaller relations that avoid the update anomalies. (For a more in-depth discussion, see (7), Chapter 14)

The CAD normalization tool is designed to allow the user to specify functional dependencies and subsequently break down or decompose the relation. Using the specified dependencies, the tool normalizes the relation to 3NF, creating new relations that are designed to reduce data redundancy. The new, normalized relations are then written to a file in a format that may be used by other elements of the data base management system.

As stated in Chapter I, the emphasis of this thesis effort is on the user-interface of the CAD normalization tool. However an understanding of the larger system that incorporates the normalization tool is required in order to evaluate and re-design the tool's human-computer interface. That is, the functions and capabilities of the other elements of the system must be kept in mind when defining the requirements and designing the interface for the normalization tool. Further discussion of the impact of the "system view" on the definition of requirements for the normalization tool is provided in Chapter IV.

Existing User-Interface

The following brief description of the user-interface of Jankus' normalization tool is provided as a point of reference from which this thesis effort was developed. The discussion includes a general view of user interaction, and a description of the tool in terms of program control and screen layout. It should be emphasized that this is not an attempt to criticize or judge the efforts of Jankus in creating a user-friendly interface. The bulk of Jankus' work dealt with the refinement of the algorithms used in the actual normalization process and in the development of the communications software necessary for using the Summagraphics Summamouse with the LSI-11. Rather, the interface description and subsequent summary of student critiques is presented for the purpose of highlighting human-computer interface issues that form the basis of this thesis effort.

Jankus developed two normalization tool prototypes, one using keyboard entry for user input, the other employing a mouse to input information. Other than the methods of data input, the main differences between the two prototypes were:

-- the mouse method used two CRT's, one for display and selection of relation attributes, the other for displaying program information to the user.

-- the keyboard method used only one CRT to display both the list of relation attributes and program information.

-- the mouse method made use of techniques such as reverse video and computer generated audio in presenting information to the user.

The prototypes were identical in the method used for controlling program flow. Presented with a series of questions requiring a yes or no answer, the user's response (either a "Y" or "N" keystroke or depressing the left or right button on the mouse) would direct the next program action. For example, in specifying the determinant attributes for a given dependency, the user would be asked to select a determinant attribute from a displayed list of entities. Selection of the attribute was accomplished by either typing the number of the desired attribute, or using the mouse to move the CRT cursor over the attribute and depressing the center button on the mouse. After selection, the user was then asked if there were any more determinant attributes for the dependency. If yes, the selection process was repeated; if no, the user was then asked to select dependent attributes for the functional dependency. Multiple dependent attributes were prompted and selected in the same manner as the determinants.

When the user responded that there were no more dependent attributes to be specified, the determinant and dependent attributes for the defined dependency were displayed on the CRT and the user was asked to verify the correctness of the functional dependency. An "N" response caused the dependency to be destroyed, followed by a question as to whether there were more dependencies to be specified. A "Y" response resulted in the saving of the dependency, followed by a query as to whether there were more dependencies to be specified. The user could continue specifying functional dependencies or quit, in which case the program automatically initiated the normalization routine. With normalization complete, the program then prompted the user for a name for each newly created relation, and echoed each relation name to the CRT. Finally the program gave the user the option to specify additional functional dependencies, or exit the program. Upon exit, the user was informed of the names of the files that contained the results of the normalization process.

As indicated in the discussion of program flow, the tool's communication with the user came in the form of single line prompts. Generally, all information was presented to the user on the upper half of the CRT, with no attempt to create functional areas on the screen for the display of different types of information. Although error messages were displayed on the CRT, very little information was provided to the user in terms of help, and no specific help function was available.

Student Critiques

No formal testing or evaluation phase was carried out for the normalization tool. However, a group of 30 Computer Systems graduate students, enrolled in a computer data base systems class, were assigned the task of using the tool to normalize a given relation. The group was asked to first specify functional dependencies and normalize the relation by hand, and then use the normalization tool (both keyboard and mouse prototypes) to carry out the task. Each student was asked to keep track of the amount of time it took to perform the task using each of the three methods and to submit any additional comments about the usefulness and user-friendliness of the CAD tools.

Since no standard questionaire was developed as part of a formal evaluation of the tool, it was not possible to perform a detailed statistical analysis of the results of the student evaluations. However, the comments provided by the students were very useful in identifying features of the CAD tool that were either confusing or not useful. In addition, the comments gave a strong indication of those aspects of the program that were something less than user-friendly.

The exact wording of the student comments about the two prototypes varied to a great exent. However, the keyboard and mouse prototype features that were most frequently criticized can be categorized into six general areas. The following is a brief summary of each of these problem areas, together with an example that illustrates the specific problem:

Usefulness. Nearly one-third of the students remarked that they found no advantage to using the tool over performing the task by hand. Several people commented that it was necessary to draw out the functional dependencies by hand before being able to enter the data using the CAD tool. One possible reason for this criticism was that the tool did not really model a familiar task for the users. The group had accustomed itself to physically drawing boxes and arrows to indicate the relationships between attributes, and this visual representation was not available using the tool.

Also, the relation that was used in the task had only a small number of attributes, making the task appear trivial in the minds of the students. Had there been a larger set of attributes involved in the relation, thus complicating the normalization process, the students' view of the tool's utility may have been different.

Consistency. Approximately one-third of the group commented that the program was inconsistent, primarily in the area of system input. For example, in some instances in order to respond to a prompt, it was necessary to type a designated key, followed by a carriage return <CR>, while in other cases it was only required to type a single key with no <CR>. As another example, a screen prompt would indicate that striking any key would result in a default to another portion of the program; sometimes the program would default, in other instances a specific keystroke was needed. Inconsistencies such as these can lead to confusion and frustration for the user.

Error Tolerance. Six of the thirty students remarked that the tool was not very tolerant of errors. In some cases, for example, when a user typed an input value that was out of range, a short error message would be displayed, and the user was given another chance to input the data. However, in the case where the user had selected all attributes of a given functional dependency, then discovered an error in one of the selections, all attributes would have to be re-specified to correct the error. For a relation with a small number of attributes, this problem would be minimal. But for a functional dependency containing a large number of attributes, recovery from such an error could be very time consuming.

On Line Help. The degree of program helpfulness (or lack thereof) was commented on by at least ten individuals. Remarks ranged from some of the instructions being confusing with not enough information, to complaints that there was no way to get shorter prompts for more experienced users. The latter comments probably stemmed from the fact that both prototypes relied on a very repetitive process in selecting determinant and dependent attributes. Once familiar with the process, speed could have been increased by allowing the user to selectively choose the extent of available prompts. Since such selection was not available, one degree of help level had to suit all users.

System Output. Many students commented on the lack of useful output provided by the tool. Part of the problem in this area could stem from a difficulty mentioned above in the area of usefulness. Although the program successfully performed the normalization task, and formatted the output for use by other modules in the DBMS, output was not provided for the user in a format so that he could readily see and understand the product of the tool.

Hardware Distractions. Comments in this area pertained mainly to some of the devices used in the mouse prototype. Approximately half of the students complained that the speech synthesizer was annoying and added more confusion to the process than help. Likewise, several people commented on distraction caused by use of the mouse. Mouse calibration was a very tedious process and users experienced some difficulty moving the cursor via the mouse. Due to a mouse scaling error, forward and backword movement of the mouse resulted in completely opposite movement of the CRT cursor. In addition, some people criticized the use of two CRT displays in the mouse prototype, saying that use of the two screens was distracting. The difficulties encountered by the individuals using the mouse prototype exemplify the problems that can arise if the available hardware devices are not carefully integrated into the human-computer interface.

In summary, the purpose of this chapter was to describe how the normalization tool fits into the larger data base management system, provide more insight into the user-interface of the CAD tool prior to the start of this thesis project, and finally to highlight some of the problem areas of the existing tool as a reference point for further research. The problem areas discussed above are not unique to software written in an academic environment. Rather, these critical areas, although often ignored by even professional system designers, must be addressed in the design of user friendly software, and are the subject of Chapter III.

III. Literature Review

Human-computer interface and user-friendliness are topics that currently receive a great deal of emphasis in computer system literature. Since this thesis effort focused on the design of a user-friendly interface for a computer aided design tool, a literature search was conducted to learn how various experts view the human-computer interface, and to examine the principles and techniques that can aid in the design of a more "friendly" computer dialogue.

The purpose of this chapter is to present a synopsis of the material researched during the course of this project. First, a perspective will be given as to how various experts view the human-computer interface, along with a discussion of how this perspective has changed during recent years. Having an understanding of what is meant by a human-computer interface and a user friendly dialogue, interface-design principles will then be discussed, principles which can help the designer create more user friendly systems. Finally, a program design and development process will be presented that incorporates these design principles.

Perspective of the Human-Computer Interface

User-friendliness, human-computer interface, and human-computer dialogue are all terms that are used in the discussion of man's interaction with computers. It is very difficult to find a concensus opinion of what is meant by these terms, because various experts have their own perceptions based on their academic areas of interest. Moran, for example, points out that technologists, designers, and cognitive psychologists all view user interaction in different ways (19:20). Technologists commonly perceive system power as the key to successful user-machine interaction, where all problems can be solved through sheer power and flexibility. System designers on the other hand, place more emphasis on the qualitative aspects of the interaction, using intuition and common sense to design a system that appears to meet the needs and expectations of the user. Cognitive psychologists differ from system designers, in that they use scientific methods to understand the complexities of user behavior during the interaction with a computer system. Of these three disciplines, cognitive psychology is having an ever increasing impact on the design of current interactive systems. More emphasis is being place on the design of systems that recognize and accomodate the individual cognitive styles of potential users. The designer's view of a system's user friendliness differs from that of a single user, as does the view from one user to the next. Failure to recognize these differences can lead to the design of ineffective, or user unfriendly computer interfaces.

Human-computer interface problems that have surfaced in the past three decades are due to a large extent to an attempt to view the human computer interface in the same light as the traditional man machine interface (3:7). In a traditional sense, man has been viewed as an "operator" of a machine, using keys and controls to manipulate it; for example, a secretary operates a typewriter through its keys, or a driver operates an automobile through its steering wheel, stick shift, etc. This "operator" point of view was applied to man's interaction with computer systems of the 1950's and 1960's. However with the advent of personal computing power and the development of interactive computing systems during the last decade, designers have come to the realization that man does not just operate a computer system, but rather he must be able to communicate with the system in order to accomplish a task.

Heckel places a great deal of emphasis on the issue of communication, stating that the writing of computer software _is_ a communication task (14). He draws an interesting analogy between the development of computer software and the development of other communication forms. For example, in the early days of the motion picture industry, the industry was controlled by engineers who took a very static approach to the way that scenes were presented. Early films were characterized by the use of a single camera, shooting from very limited angles and long distances. Emphasis was placed on the mere presentation of a scene, rather than on _how_ the scene could be presented to have a greater impact on the audience. It was not until artists entered the industry, such as D.W. Griffith and his classic Birth of A Nation, that the industry shifted from an engineer's craft to that of an artist. Using filming techniques such as the close up, the fade, and the cutaway, artists were able to transform the motion picture industry into one that truly communicated with an audience.

Heckel contends that this same sort of transformation is required in the design of computer software, if systems are to be effective. Designers' perspectives must change so that emphasis is placed on not just what the software does, but how it does it.

Having looked at how some experts may view the nature of the human computer interface, it would seem appropriate then to define what it is meant by a "friendly" user interface. Unfortunately, coming up with such a definition is no easy task because of the extent to which expert opinion varies on the subject. Too often, when asked to define user friendliness, people may respond with, "I don't know what it is, but I know it when I see it" (24:75). Unfortunately, this type of response is quite common from both system users and designers alike, and provides an indication of how difficult it can be to design a friendly system. Raduchel gives a better insight when he defines a user friendly system as one that helps produce accurate solutions in less time and at less expense than alternative systems (24:77). Simpson takes the definition somewhat further when he describes a user friendly program as: "...one with features that acknowledge human factors...one that is easy to use, tolerant of operator errors, easy to learn, and acknowledges that human beings are imperfect creatures" (30:2). Based upon this definition, and those of other authors referenced during this project, designer awareness and acknowledgment of human factors is paramount in the design of interactive systems. Some of the elements that designers must consider in the area of human factors include: ease of learning, ease of use, reliability and robustness, tailoring and productivity (28:8-11). These and other elements will be discussed in greater detail in the next section on design principles.

Before concluding this discussion of the perspective of human computer interface and user friendliness, it is important to point out that the development of a user friendly interface can be very costly in terms of both time and money. In a recent study of twenty-two European Interactive Business Applications (IBA), nearly sixty percent of the code written for the average IBA was devoted to managing dialog with the user. The average 17,500 lines of dialogue management code increased both the cost and development time for the systems (5:14). Because of this cost, an argument can be made that in some cases, extensive user friendliness in the design of a system may not be justified. Examples of such cases might include: simple programs designed for the use of one or two people, programs without serious error consequences, or programs designed to answer a single question or a one of a kind problem (30:18). In these instances, system developers may have to ask how much human factoring design will improve system performance, and if this improvement will counterbalance the additional development costs.

Having looked at what is meant by the human computer interface and having attempted to define what is meant by a user friendly system, it should be apparent that there are no simple definitions, nor is there a cookbook approach that designers can use in developing friendly interactive systems. Designers must have an awareness of human factor issues, and if possible, use a set of design principles or guidelines that can help in the design of more friendly systems. A possible set of guidelines are the subject of the next section.

Design Principles

Current software design literature presents a wide variety of design principles that designers may use in developing a computer interface. Since there is no cookbook approach to the design of user friendly software, the designer must internalize general design principles, and then apply them to specific design problems (30:19). The application of these principles and rules of thumb can in no way guarantee the quality of the interface. However, an awareness of various guidelines can be an aid to the designer in developing more friendly systems.

In conducting the literature search for this thesis effort, it became apparent that there is a wide range of opinion as to the appropriate guidelines that designers should consider. For example, Heckel outlines thirty elements of friendly software design that range from knowing the subject for which the system is being designed, to keeping the design simple (14). Similarly, Woffinden provides a comprehensive listing of design principles proposed by six different authors, each author having a different perspective on significant design guidelines (33:18). Even with this wide range of perspectives, there does appear to be general agreement on the basic principles that must be considered. Software designer Henry Simpson offers a series of twelve design guidelines which are representative of the diverse principles found in current literature (30:46-60). The following is a brief discussion of these guidelines.

(1) Define the User. Just as an author must identify who it is he is writing for, so, too, the software designer must have an awareness of his intended audience. Only by identifying the needs and capabilities of the prospective user can the designer develop a system that is tailored to the individual user, or group of users. Carey provides a detailed discussion of the importance of recognizing user differences in interface design (4:12-20). Among the factors that Carey says must be considered are: the nature of the task to be performed and the associated mental task model of the user; the user's cognitive model of how the designed

system is to work, and what the system can do for him; and the degree of exposure that the user will have with the system. The designer must be aware of these individual user differences, and more importantly, design the system so that it accomodates operator growth as the user becomes more familiar with the system.

(2) Anticipate the Environment in Which the Program is to be Used. The environment in which a system will be used influences many aspects of a systems' design. Factors that should be considered include the noise and lighting level of the work environment, the physical size of the users work space, and the extent to which a user may be interupted while using the system. These factors may impact the selection of hardware devices used in the interface, as well as determining how flexible the system should be in allowing the user to pick up and continue work after an unexpected interuption.

(3) Give the Operators Control. This is a very significant design principle in terms of its human factors implications, and may have a great influence on the users' acceptance of a new system. Heckel points out that the more control that the user is allowed to exercise, the better the user will feel, and the better he will like the program (14:58). Rather than being told what he can or cannot do when operating a program, the user should be given the impression that he, not the computer is in control. Likewise, instead of having the system overwhelm the user with too much

information, too quickly, the user should be able to manipulate the program at a comfortable level. Shneiderman emphasizes this aspect of user control in his discussion of "direct manipulation". The central ideas of direct manipulation include making the objects of interest visible to the user, providing rapid, reversible actions and replacing complex command language syntax with direct manipulation of the object of interest (29:13-25).

(4) Minimize the Operators' Work. System developers may loose sight of this seemingly obvious design principle through a poorly designed system interface. Heckel comments that, "The most important thing to know regarding your user is that he is not interested in using your product. He is interested in doing his work, and your product must help him do it more easily" (14:27). Woffinden comments that, "...the designer should create a tool from the conceptual view of an assistant to the programmer, with the goal being to have the user of the system view the result not as a computer system, but as an assistant" (33:34). Minimizing the operators' work applies to any software product, but is particularly relevent when applied to the design of a computer aided design tool.

(5) Keep the Program Simple. This guideline, similar to the previous one, may at times be ignored in the design process. Andriole addresses simplicity in the user dialogue when he states that designers may avoid using simple dialogue

for fear that their work may be labled as unsophisticated (1:27). Although it is important to recognize the difference between a simple and a simplistic system, fears such as this can often result in hard to use systems.

(6) Be Consistent. System consistency is an important user concern, as was illustrated by the comments of those students who used the existing CAD normalization tool. A lack of consistency can make a system very difficult to learn and can lead to a great deal of frustration on the part of the user.

(7) Give Adequate Feedback. If one is to accept the premise that the human computer interface is characterized by user system communication, then providing adequate feedback is a critical design consideration. As Simpson points out, human beings live by feedback, and an effective dialogue can only occur if a useful two-way conversation is created (30:53). Any action that the user may take, whether correct or incorrect, should result in some type of response from the system. The manner in which the system responds to the user is an important design concern, and worthy of further discussion.

Dean maintains that designers should apply psychology when writing messages for the user, attempting to think as the user is going to think. To achieve this, the designer may want to playact, to evaluate messages for usability prior to coding (8:430). Andriole suggests that in order for user-system dialogue to be effective, the dialogue should:

(a). be clear, concise and courteous

(b). tell the user only what is necessary to continue the interaction

(c). be as specific as possible

(d). be directly and immediately useful

(e). use humor with caution

(f). use words that are short, easily interpreted and unabreviated

(g). use sentences that are brief, simple, in the affirmative and active voice (1:32)

These user-system dialogue characteristics closely parallel commonly accepted features of effective writing or speaking, again alluding to the fact that the creation of effective feedback is a communication task.

<u>(8) & (9) Do Not Overstress Working or Recall Memory</u>. These design guidelines are based to some extent on Miller's classic research of human capability to process information (18:81-97). Humans are limited in their ability to hold information in short term memory, and in their ability to process different pieces of information. System designers must be aware of these limitations when creating screen layouts, designing menus and displaying information to the user. For example, if the user is required to use data that requires several screens to display, he should not be forced to remember the information contained on all of the screens. Rather he should be able to page or scroll through the infor-

mation to access what he needs. Another example pertains to the design of menu displays in a menu driven program. An inexperienced user should not be expected to remember the function of all menu items; rather he should have some type of help facility available to him to refresh his memory on menu-item functions. Over tasking of user memory can once again lead to user frustration and dissatisfaction with a system.

(10) Help the User Remain Oriented. The goal of keeping the user oriented is closely related to that of providing adequate feedback, since both are concerned with keeping the user informed about what is going on in the program and where he is at any point in time. Carey draws the analogy that upon arrival in a strange city, a person may require a physical map of the area to help navigate through unfamiliar streets. As the individual becomes more familiar with the city, he becomes less dependent on the physical map and develops his own "mental map" of the city layout (4:15). In a similar fashion, a new or inexperienced computer program user needs a "map" of where he is in the program, relying on this information until he is able to develop his own mental picture of how to maneuver within the program. Simpson suggests that this mental map can be developed by giving the user thorough documentation as well as ample prompting within the program (30:57). Possible techniques that the designer can use in enhancing the development of a mental map are the

use of reverse video to highlight selected items in a menu, or changing the border region of an area of the screen when it becomes active (28:47). Whatever techniques the designer choses, he must ensure that the user is aware of his location in the program at all times.

(11) Code Information Appropriately (Or Not At All). Information coding can be interpreted as the form in which the computer presents information to the user (30:58). The program designer may have several types of codes at his disposal. Some of the types presented by Simpson include: color coding, sound coding, alphanumeric coding, shape or size coding, and brightness or flash rate coding (30:122-123). The designer must select the form of coding that is appropriate for the type of information he wishes to convey to the user. For example, if the user attempts to perform an operation that has serious, irreversible consequences, the designer may want to use an audio tone to warn the operator of the impending consequences. Or if the user must access a great deal of data, the designer may choose to encode the data in graphical or tabular form rather than presenting it in raw form. Selection of the appropriate form of information coding depends on the designer's previous success at identifying the user and defining the user's needs.

(12) Follow Prevailing Design Conventions. Taken at face value, this design principle appears to discourage innovative efforts by the designer to experiment with the user-

computer interface. Rather, the intent of this guideline is to ensure that designers are aware of the fact that there are certain prevailing conventions concerning the way that operators are supposed to interact with computers. Using a rather simplistic example, operators are accustomed to using the return key on the keyboard to transmit information to the computer. If the designer should deviate from this convention and use the escape key for the transmit function, users may well have a difficult time adapting to the change.

In deviating from accepted conventions, designers must be confident that the advantages of the change, in terms of user-friendliness, far out-weigh the difficulties that users may experience in breaking old habits. System designers at Apple Corporation for example, must have taken this into consideration when they introduced the Lisa and MacIntosh computers. The use of the mouse for user-system interaction was a revolutionary concept, in a market where the keyboard provided the accepted means of communication. Although typical design changes will not be as dramatic as in this example, the designer must acknowledge existing conventions and carefully consider the impact of deviating from these standards.

In summary, the set of design principles discussed above are by no means an exhaustive list of the areas that designers must consider in developing interactive programs. However an awareness of these basic guidelines will aid the designer as he begins the program design and development process.

Program Design and Development Process

One could argue that the design and development of computer aided design tools, tools that rely heavily on the effectiveness of their human-computer interface, should be no different than that for any other type of software product. In a book entitled Developing Expert CAD Systems, author Vivienne Begg contends that:

> It is well known that bad human computer interface design can cause intense frustration for users. However, it seems the lessons of ten years of research into dialogue design for interactive computer use have not always been learned by those who design CAD systems. In the author's experience the most frequent complaint made by CAD users have been about menu systems, on line documentation and error messages, where the application of a few simple principles of software design were all that were needed to render the system far more user-friendly (2:34).

A user friendly system can only be created when human interface issues are raised early in the design process, rather than being considered as an after-thought. Software developer Richard Rubenstein amplifies this point when he says,

> ...adding new things to system design usually leaves the hard to use part as it was, and makes the system more complex and hard to use...ease of use is an emergent property, a broad consequence of design, not a specific feature (28:9).

In the field of software engineering there are a wide variety of models that are used to describe the software design and development process. Peters discusses several of these models ranging from the waterfall model of the software development cycle, to hybridized life cycle models (23:12-17). In examining the various software development models, a valid question can be raised as to whether any of these models provide a viable approach to software development, in cases where the human computer interface is of primary concern. Software developer Henry Simpson suggests that popular approaches such as top-down or bottom-up design are both "inside-out" approaches, focusing first on the inside of a program and only later on the outside. As an alternative, Simpson proposes a "top-down, outside-in" approach, in which the focus is first placed on the interface, then on the design of a program which produces the interface (30:62-63).

The following discussion is centered on Simpson's eleven step program development process. There are many similarities between this approach and other commonly accepted development models, with the exception that more emphasis is placed on human interface issues. Because several of the development steps parallel the design guidelines discussed in the previous section, not all steps will be discussed in the same degree of detail.

(1) Define the System Objectives. During this stage of development, the designer must perform a systems analysis to gain an understanding of the problem to be solved. Having done this, he must then define what the system is to do. At this stage, the objectives are stated in rather general terms, with further refinement to follow in step three.

Without a clear understanding of general system objectives at the outset, later stages of the development process will be complicated, or worse yet, at the end of the entire development process, a working tool may be produced that doesn't accomplish what the user had intended.

(2) Define the System Users. As discussed in the previous section, the designer must identify the intended system user. User characteristics that should be considered include: education level, computer system familiarity, frequency of system use, and knowledge level of the task to be accomplished. The less diverse the intended audience, the easier the design task for the system developer. However, even in dealing with a limited group of users with similar characteristics, the designer must accomodate user growth, recognizing that operator needs will change as familiarity is gained with the system.

(3) Define the System Functions. Having defined system objectives, and identified the intended user, the designer must then address the specific functions of the system. This stage of the development process involves preliminary design work, specifying the functions needed to accomplish the system objectives. However, the specified functions are still at a high level, and lack sufficient detail to begin implementation.

(4) Plan the System Modules. At this stage of the development process, the designer determines the software

modules required to accomplish the functions specified in step three. In some cases, the system functions may translate directly into system modules. However, depending on the complexity of a given function, several modules may be required to accomplish a specific function. In this development step, Simpson suggests the use of a top-down design approach, breaking the design problem into smaller pieces. Each module is subsequently broken down into sub-modules, with the process continuing until the subroutine and individual code statement level is reached. Once the lowest design level is reached, the designer should then draw up a list of human-computer interface requirements for each sub-module. This set of requirements will be used in step six, where the actual interface is designed.

<u>(5) Select the Hardware Configuration</u>. The steps of the design process up to this point have been independent of any hardware considerations. The designer has defined system objectives, identified the user, outlined system functions, and broken the problem down into smaller pieces. At this point it is necessary to consider the hardware that will be used to implement the system. The designer's hardware considerations must include:

(a). the memory and processing speed characteristics of the machine on which the system will run.

(b). the secondary storage capability of the target system.

(c). selection of appropriate input and output devices among those that are available.

Given these and possibly other hardware considerations, the designer must insure that further development can be supported by the available hardware.

(6) Design the Human-Computer Interface. At this point in the development process, the designer must deal with the specifics of how the operator will interact with the computer. Using the interface requirements generated in step four, the designer must deal with three essential aspects of the interface design: system output, system input and program control. The following discussion deals with each of these design aspects in greater detail. Due to the volume of information that is available in current literature pertaining to each of these areas, however, the discussion will be limited to only a few general issues that relate to each aspect.

System Output. System output deals with how the computer displays information to the operator. Many display options are available to the designer, ranging from the use of strictly textual displays to the use of modern graphics techniques to display information in pictoral form. Raeder points out some of the advantages of graphical displays, advantages that include: providing random versus sequential access to information; providing a multi-dimensional rather that a one dimensional view of information; offering a greater transfer rate of information to the user; and providing the capability to use real world objects to illustrate abstract ideas (25:12).

Regardless of how the designer may choose to display information to the user (graphically, in textual form, or a combination of the two), a recommended technique for information presentation involves the use of functional areas on the display screen. Heines, in a study of students using a system for computer aided instruction, found that the use of functional areas contributed clarity, consistency and continuity to the screen design. Consistent use of general screen areas for certain types of information helped students maintain their orientation, minimized the effort needed to interpret instructions, and allowed students to concentrate on the subject matter rather than on the mechanics of the program (15:17). Norman also found that the use of functional areas aided in the prevention of description type errors that users may commit (20:254).

In addition to the use of functional areas in screen layout, various visual attributes can be effectively used to distinguish different types of information. Rubenstein discusses several possible display attributes including blinking, brightness, size, color, shape, and font, any of which can be used to call the user's attention to something on the screen. However, the designer should use good judgement in combining or using these attributes to avoid overwhelming the user. Indiscriminant use of these attributes may add confusion rather than clarity to the display (28:171).

System Input. The interface design aspect of system input deals with the manner in which the operator enters data into the computer. The specific input device used in the interface is obviously dependent upon the type of hardware device available on the target system. However, Giloi sights a trend in modern programming in which software packages are device independent. That is, input devices should be characterized by their function rather than by technical idiosyncrasies (12:200). Foley and Wallace define the use of "virtual" input devices that consist of: a selector, used as a pointing device; a locator used as a positioning device; a valuator used as a device for the input of constants; and a button used for the input of commands (11:462-471). Giloi states that using this virtual device concept,

> ...the application programmer may concern himself only with these virtual devices, regardless of whether selection is performed with a lightpen or by placing a cursor, whether location is performed through a cursor positioning device or by pen track, whether button is a control key or a light button etc (12:200).

Taking this approach, a software interface would be required between the service routines for the virtual device and the physical devices chosen for the interface.

In selecting a specific physical input device, the designer must evaluate the merits of a given device for the specific application being designed. Numerous studies can be found in current literature that address the ergonomic considerations of various input devices. (See Card(3), Rubenstein(28)) The following is a brief discussion of some of the advantages and disadvantages of some of the devices considered for use in this thesis effort.

--Light Pen. The light pen is a selection device that was very popular in early graphics applications. It allows the operator to point directly at an item or area on the screen with relatively fine resolution. Due to the method with which the pen's screen position is determined, the light pen can only be used with refreshed picture displays versus storage tube displays. Common uses for a light pen include menu selection, tracking and graphics input. When an application calls for the use of a light pen over an extended period of time, operator arm fatigue must be considered. In addition, the light pen, like other devices to be discussed, cannot be used for direct input of data. Therefore it may be necessary for the user to transition back and forth from the light pen to the keyboard when data input is required.

--Touch Screen. A touch screen, similar to a light pen, allows the operator to point directly at an item or area on the screen. However, it is not possible to achieve as fine a resolution as can be obtained with the light pen, and user arm fatigue may also be a problem over extended periods of time.

--Mouse. A mouse is currently the most popular device used for system input (26:112). It provides a very rapid and accurate method of pointing to items on the screen, and does not have the drawback of user fatigue since the device is used on a flat surface. In a recent study conducted by Card, Moran and Newell, a mouse was preferred over a joystick, cursor keys or text keys in performing a task of text selection. Cursor positioning time was significantly faster, operator error rate lower, and the rate of movement of the mouse was nearly maximal with respect to the information-processing capabilities of the eye-hand guidance system (3:229-256). However, use of a mouse does require a larger working area, and some people using a mouse for the first time may have difficulty coordinating cursor movement with the mouse while watching the results on the screen (9:49).

--Voice Input. Voice interaction appears to be a trend of the future for system input. A primary advantage of voice input over other methods is that its use allows interaction with the system, and at the same time frees the hands to perform other tasks. However, because an auditory signal can exist only in real time, processing from text to speech must occur in real time for the audio to be meaningful (28:83-84). With present technology, speech systems are only able to recognize some words with fair to good accuracy, and are limited to a relatively small vocabulary. However, as technological advances are made, voice input may be a viable option, because it is a natural companion to many cognitive and motor operations (1:86).

Program Control. Having looked at the aspects of system output and system input, a third interface design consideration is that of program control. Program control refers to how the operator controls what the computer does, and can be generalized into two main methods: computer initiated dialogue and operator initiated dialogue.

Computer initiated dialogue can take several forms, including question and answer, and menu selection. In a question and answer format, the system queries the user at each stage of the program for direction as to what action to perform. This is a very simple technique, however, it offers no shortcuts and requires the user to answer every question every time the program is used. A menu selection format, on the other hand, allows the user to choose from a range of options displayed in a menu. Rather than being forced to answer a series of questions to get from one point in the program to the next, the user can make a choice and move along with greater speed than is afforded with the question and answer technique. Both of these methods are better suited for programs that are highly structured in nature, with a limited number of options from which to choose.

User iniatiated dialoque is characterized by the use of a command language syntax in which the operator types a command or series of commands to direct movement within the program. This method allows greater flexibility in moving through a program, and is better suited for systems that have a more unstructured task to perform. However it does require the user to remember specific syntax.

In making a choice between computer initiated and operator initiated dialogue, the designer must have a clear understanding of both the task to be performed and the intended user. Andriole points out that experienced operators who use a system on a regular basis can benefit from well engineered interactive dialoque based on function keys or a command syntax. However, even experienced users can forget functions and commands (1:27). For this reason, menu based dialogue is a common technique used in interactive systems. Raeder comments that: "...menus make possible human-computer interaction that is simple enough for novices, yet fast enough for experts. The primary benefit of menus versus stated commands is that a menu does not require the user to memorize syntax. Another interesting aspect is that a menu-based system facilitates system exploration"(25:17).

Whatever method of program control is implemented, the designer must ensure that it is compatible with the other aspects of the interface design. The three aspects of the interface design, system output, system input and program control, must all blend together in forming an effective, user friendly interface.

<u>(7) Design the Data Structures and Files</u>. Having designed the human-computer interface, the next development step is to determine the necessary data structures and files. These items contain the information to be manipulated by the user and have an impact on the ease with which the user can use the system. Factors such as access speed and ease of data modification can be influenced by the data structure and file design. In a well designed interface, the data and file manipulation task can be made transparent to the user, so that the operator can accomplish the task for which the tool is designed, without having to worry about the details of data manipulation.

<u>(8) Prepare the System Specification</u>. One could argue that this step of Simpson's program development process is out of sequence and should have occured earlier in the process. However, in the context in which Simpson is using the term "system specification", he is referring to a stage in which all of the design work previously accomplished is compiled. This compiled information provides a complete description of the system, which is then used for final design review before code is developed or hardware acquired (30:72). It is safe to assume that system specifications would have been previously addressed, with this being the final review before coding begins.

<u>(9) Write the Program Code</u>. After the design work of

the previous eight steps is completed, the task of program coding begins. Various strategies may be used in the coding process, the most logical being to code and test a central module, and to continue the coding and testing down through the modules of the design. In large projects, where many programmers are involved, several modules may be coded and tested simultaneously, and then linked together at a later time.

(10) Document the Program. Documentation may be broken into two general types: system documentation written for people who will have to maintain the program; and user documentation for the operators of the program. System documentation will consist of any items produced in the design phase such as SADT's, structure charts, or pseudo-code, as well as any embedded documentation such as module headers or program comments, all of which may aid in changing or maintaining the program.

User documentation will consist of both off-line and on-line information that will help the operator use the program. Off-line documentation will be in the form of user's manuals that the operator can use to get detailed information about the system. On-line documentation is that information which the operator can obtain while actually using the system. What on-line information is available to the user, and how he receives it, should have already been planned for in the interface design phase of the development process.

(11) Test the Program Program testing, rather than just occurring at the end of the development process, is an on-going process that occurs throughout the coding phase. Testing should also have been addressed early in the design stages, in order to establish a viable testing plan. Tests should be conducted to ensure not only that individual modules operate correctly, but also to ensure that the system requirements outlined early in the development process are satisfied.

The eleven step process just described is only one model or perspective of how the program development process should work. As mentioned earlier in this section, there is very little difference between it and other more widely accepted views, with the exception that more emphasis is placed on human-computer interface issues early in the process. This or any other process model, however, cannot be expected to be implemented in the sequential fashion in which it was presented. Peters makes this point when he says:

> The above descriptions (models) are overly simple but they capture the basic thrust of the development life cycle, which presents software development as a set of sequential, interdependent steps resulting in a product. This would not be a particularly hazardous view if we were dealing with a tangible product, operating time frame, stimuli, and environment; but in software development we are dealing with logic. Things do not work in this simple, sequential way. Problems crop up in later steps that affect the results of earlier ones, and so on. One contention is that the development process could work this way

> if we diligently completed each phase before going on to the next. Although this idealism sounds reasonable enough, it does not reflect what happens in any engineering and scientific fields (23:13).

Software development models do provide a viable framework from which systems can be created. The next chapter consists of a discussion of the process used in developing the CAD normalization tool for this thesis effort.

IV. CAD Normalization Tool Development Process

In light of the design guidelines discussed in Chapter III, a complete redesign of the existing system was deemed necessary to attain a more user friendly CAD normalization tool. Although software modifications could have been made to add user friendly aspects to the existing code, such an approach would have been contrary to the precept that ease of use is a broad consequence of design rather than a specific feature that can be added to a system (28:9). Therefore a complete redesign was accomplished using the eleven step process described in Chapter III. The remainder of this chapter describes the development process as it applied to this thesis effort.

(1) Define the System Objectives

The objective of the CAD normalization tool is to provide the data base administrator with an interactive system through which he can specify functional dependencies that exist among attributes of a given relation. Using the specified dependencies, the tool should then perform a non-loss decomposition of the existing relation, normalizing it to third normal form.

As previously discussed in Chapter II, the normalization tool is only one component of the larger data base management system. As such, it depends on the output from the DBMGR component, which allows the DBA to specify the domain and relational information used in the normalization process. The normalization tool must be capable of reading this information in the format provided by DBMGR, and likewise output the new relational information in a format compatible with other elements of the DBMS.

(2) Define the System User

The designated user of this tool is a data base administrator, or a member of his staff. These individuals are computer professionals who have a thorough knowledge of the normalization process, and of the computer system on which the tool is to be implemented. The following assumptions were made concerning user characteristics, assumptions that influenced subsequent stages of the design process:

(a) As computer experts, the target users have a working knowledge of the system file structures and operating system commands.

(b) The system users will only be required to use the normalization tool on a periodic basis. That is, although they are familiar with the normalization process, the users may have extended periods of time between use of the tool.

(c) Speed and accuracy in performing the normalization task are of prime importance to the user. As the DBA regains familiarity with using the tool, he should not be burdened with excessive prompting or help features.

(d) Because the size of the relational files used by the DBA may be large, and due to the dynamic environment in which the tool will be used, the user can be expected to experience interruptions during the course of a normalization session. Therefore the system must enable the DBA to stop work at any point in the process and resume at a later time.

(3) Define the System Functions

Considering the previously defined system objectives and user characteristics, the normalization tool should provide the following functions or capabilities. The tool should:

(a) Allow the user to specify the name of an input file containing the relations to be normalized.

(b) Read the relational information from an input file previously created by DBMGR.

(c) Display the names of relations contained in a designated file, indicating their normalization status and the existence of any previously specified functional dependencies for each relation.

(d) Allow the DBA to select the relation that he wants to normalize.

(e) Display the list of attributes comprising the selected relation.

(f) Display a relation's functional dependencies that may have been specified during a previous normalization session.

(g) Allow the user to edit any existing functional dependencies, or to specify new dependencies.

(h) Give the user the capability to view any or all specified dependencies at any time during the editing process.

(i) Perform a non-loss decomposition of the current relation once the user indicates that all functional dependencies have been specified.

(j) Allow the user to choose a name for each new relation created as a result of the decomposition.

(k) Allow the user to quit at any time during the editing process, saving the current dependencies for use during a later session.

(l) Allow the user to specify the name of the output files used for storage of new relational data and any partially specified dependencies.

(m) Provide the capability to print a hard copy of all relational information at the user's request.

Given these basic system functions, a structured analysis and design technique (SADT) was used to graphically portray the upper design levels of the proposed system. This technique is the preferred method at AFIT, and proved to be a helpful tool in conducting the functional analysis prior to actually laying out the system modules.

(4) Plan the System Modules

Continuing with the development process, the functions outlined in step three were used to derive system modules. A top down approach was used, taking the basic functions and breaking them down into logical pieces. Structure charts were used to specify the modular characteristics of the system design. Chapter V provides a more detailed look at the system modules, and structure charts are found in Appendix E.

At this stage of the development process, it became rather difficult to strictly adhere to the sequential steps of the development process. As discussed in Chapter III, an orderly flow through each step of the process is certainly a goal to strive for, but is not always attainable due to the nature of the software development problem. For example, in designing the system modules, it was found necessary to at least consider the type of user interface that was desired for this application, and the type of hardware that would be used in the interface. A reasonable argument can be made that the next several steps of the development process will often occur simultaneously, rather than being treated as distinguishable steps in a sequential process. This situation is alluded to by Simpson when he states that his eleven step development process should not be regarded as "...a rigid formula that you must follow without variation. Rather, look upon it as an ideal. You must adapt it to fit your working style and the design problem that you are attempting to solve" (30:63).

(5) Select the Hardware Configuration

The Digital Engineering Laboratory System "A" , an LSI-11/73 running the RT-11 Version 5.1 operating system , was selected as the host computer for the CAD normalization tool. The primary reason for this selection was the fact that this system could support the use of a wider range of peripheral devices as part of the man-machine interface. Internal memory capability of the LSI-11 was recognized as a possible limiting factor as the CAD tool software was expanded. Memory limitations, as will be addressed in Chapter VI, did in fact impact the final system design. However, the decision was made to implement the tool on System "A" due to its peripheral device support.

Peripheral input devices considered for use in the normalization tool included a mouse, a joystick, and a graphics tablet. It was hoped that a light pen could be used in the project. However, the pen was not made available by suppliers until late in the effort. Other available hardware included a speech synthesizer, and three display screens, one for normal CRT text display, one with high resolution graphics and a third with medium resolution color graphics. Considering these devices, the decision was made to implement the mouse as a selection device in concert with the normal and monochrome graphics monitors for system output. The following is a brief discussion of why this hardware was selected over the other available devices.

In terms of graphical input devices, there are three basic functions that these devices may provide: pointing, positioning and drawing (9:46). The selection of a particular input device to a large extent depends on user preference and on how well a particular device is suited to a specific application. For this particular application, it was felt that the primary function to be accomplished was one of selection, or pointing. The main objects to be manipulated in the normalization tool were relation attributes, objects previously created in DBMGR. Attributes had to be selected, in order to form functional dependencies. Initially, use of the graphics tablet was considered, because it would afford the DBA the opportunity to actually draw the functional dependency diagrams, much in the same manner as he would by hand. However, it was felt that the user's task could be simplified by having the tool simultaneously generate dependency diagrams on the CRT, as the DBA <u>selected</u> the component attributes. In this way, the DBA would still have the advantage of having the information displayed in graphical format, yet be free of the manual task of actually drawing the diagrams. Given that the user would select applicable attributes, the mouse offered the greatest advantages when compared to the other available devices. The joystick could have been used. However, because it was a switch-activated type stick, it could not offer the freedom and precision of cursor movement that could be provided by the mouse.

The graphics terminal was selected as the primary output device, with the standard text CRT used as a secondary screen. It was hoped that the native graphics mode of the graphics terminal could be used in generating the dependency diagrams. However, as will be discussed in Chapter V, system memory limitations prevented this implementation. Use of the color graphics monitor and the voice synthesizer were considered. However, it was felt that for this application neither device would significantly enhance communication between the tool and the user.

(6) <u>Design the Human-Computer Interface.</u>

The design of the human-computer interface involved consideration of the three aspects of program control, system output, and system input. The following discussion deals with each of these aspects as they relate to the normalization tool, and describes why the selected control,output, and input methods were chosen over others that were considered. The system descriptions will be in rather general terms, with specific information being given in Chapter V.

Considering the tasks to be performed by the tool, a menu driven method of program control was selected as being best suited for this application. In using the tool, the DBA performs the two relatively structured tasks of editing functional dependencies, and then executing the actual decomposition of an existing relation. In accomplishing these two tasks, there is a rather limited set of functions that must be performed, making a menu driven system feasible. A question and answer format or an operator initiated dialogue using a command syntax were also considered. However, there were several factors which favored the use of a menu format over these other techniques.

First, a question and answer format, although the simplest of control techniques, was considered to be too slow and too simplistic for the target user. The user would be forced to go at a pace dictated by the question format, a pace that may be comfortable when using the tool for the first time, but one that could become irritating as the DBA gains familiarity with the system. Several data base students, for example, who used the existing normalization tool, commented that the question/answer format was too slow in specifying a large number of functional dependencies.

An alternative control method considered was that of a command syntax dialogue. In some ways, this method was suitable since it would allow the user greater flexibility in executing the desired functions, and would facilitate greater speed when the tool was used by an experienced user. A factor that had to be considered, however, was that the tool would most likely be used only on a periodic basis. As was pointed out in the previous chapter, even an experienced user who only occasionally uses a system has a tendency to forget a command language. Considering these factors, a menu technique was selected for the normalization tool. A two layer menu scheme was devised, in which the DBA is given the opportunity to select the tool functions. The menu layers and specific functions will be described in Chapter V.

Having selected a menu-driven format, the next task was to design how the menus and other information would be displayed to the user. Various screen designs were drawn by hand and evaluated to determine which provided the most desirable format for presenting information to the user. This design method, recommended by design experts of interactive systems, lets the system developer experiment with such factors as area shapes, area location and area boundaries before actually developing system code. In all prospective layouts, distinct functional information areas were planned, in an effort to provide consistency for the user, and to help the user remain oriented while using the system. Also, before deciding on the final screen design, the designer of a second AFIT data base design tool was consulted, to see if a common format could be developed for use in the two CAD tools (10). It was felt that such commonality would add greater consistency to the various design components, and hopefully simplify the user's task in going from one tool to another.

The selected screen layout is shown in Figure 3. The menu area appears on the right side of the display. A second functional area occupies the bottom three lines of the display giving information concerning the current menu item selected, prompting of possible actions for the selected menu item, and appropriate error messages for the user. A third functional area, located in the lower half of the screen, serves as the primary data display region and is used for the display of both relation and relation attribute names. The remainder of the primary screen is used as either a workspace while in the edit mode, or as a secondary message display area while in any of the other modes of operation. A final functicnal area occupies the entire secondary CRT. This area is used to display help information upon user request, and is alsc used to show the user any previously specified functional dependencies. The use of all of these areas will be discussed further in Chapter V.

The third interface design aspect, the way in which the user enters data into the computer, is provided by a mouse and through the use of the terminal keyboard. The mouse is

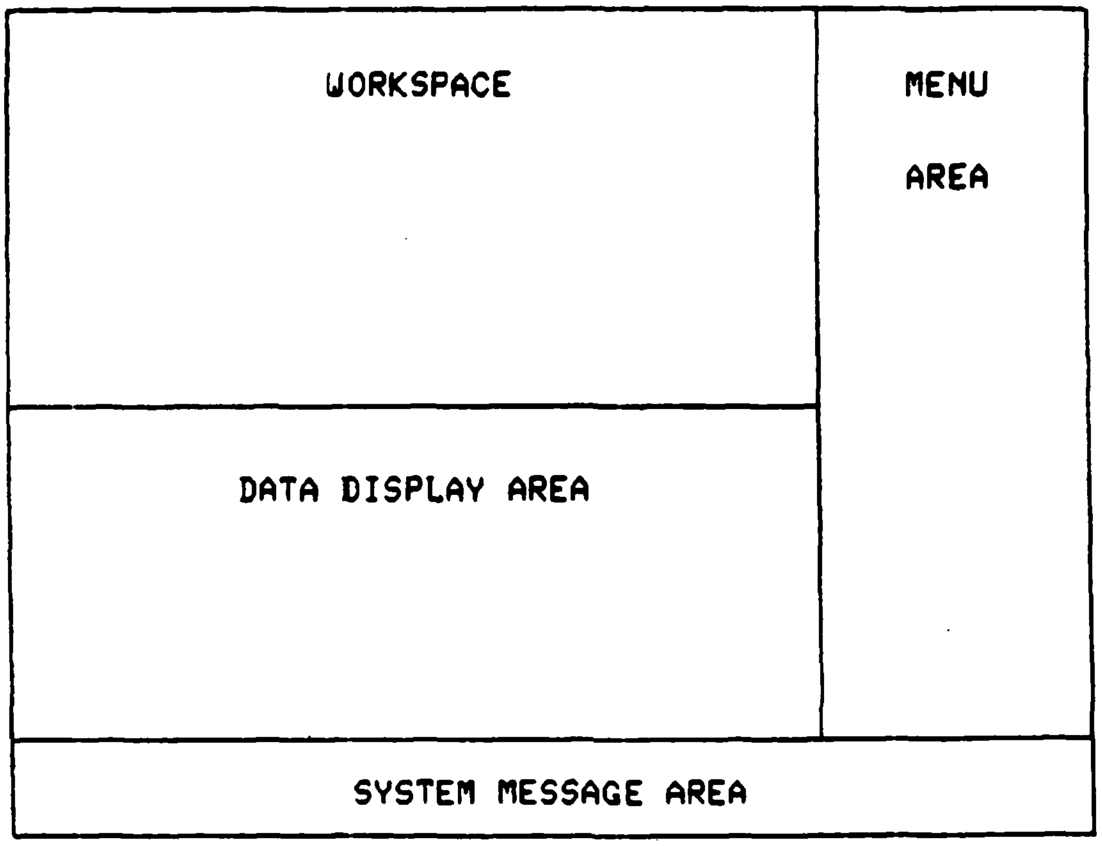


Figure 3. Functional Screen Layout

the primary input device, used to make all menu, relation and attribute selections. Keyboard inputs are minimized to prevent unnecessary user transitions from the mouse to the keyboard. Such transitions can be distracting and awkward for the user. Keyboard inputs are only required at three distinct phases of system operation as will be discussed in Chapter V.

(7) Design the Data Structures and Files.

To a large extent, the data structures and files used in this effort were pre-determined, due to the fact that code from a previous design was to be incorporated into the system. For example, the existing modules that perform the nonloss decomposition of relations use domain and relation data that is stored in a linked list structure of records. Each relation is comprised of linked structures that contain attribute information for a given relation. To ensure that new software was compatible with existing normalization tool code, and in the interest of developing modules that could be used in other applications which use the same domain and relation data, the data structures were not changed in this design effort.

Similarly, the formats used in input and output files used in this effort remained unchanged from those used in previous applications. This was necessary once again to ensure that the input and output data from the CAD normalization tool was compatible with other components of the data base management system.

(8) Prepare the System Specification

As discussed in Chapter III, this phase of the development process involved reviewing the information developed in the previous seven steps, to ensure that the design of the system to this point was as complete as possible prior to beginning the coding phase. A formal specification was not drawn up. However, all design work was reviewed and modified where necessary to ensure that all system objectives were satisfied.

(9), (10), (11) Write, Document and Test Program Code.

The final three program development stages were accomplished concurrently, and are briefly discussed together. System coding was accomplished using the Whitesmith C programing language. C is a very versatile language and is also the standard used in the Digital Engineering Laboratory. A top down approach was used in coding the system modules. This method proved to be very compatible with the two-layered menu design of the system. For example, the main module, in addition to performing system setup and initialization, called subroutines which constituted the menu items of the system's main menu. These subroutines were developed and tested prior to moving to the next lower level. Likewise, the subroutines made calls to other routines that corresponded to menu items of layer two. Coding and testing modules in this manner continued through to the lowest module level of the system. Code documentation, in the form of AFIT/ENG standard module headers, was included in all program code as modules were written. Header information proved to be helpful during program development as coding changes were made, and serves as an aid for future system modifications.

Having looked at the total program development process used in this effort, the following chapter will look at specific design aspects of some of the more significant program modules, and offer a better understanding of how the normalization tool is used.

V. CAD Normalization Tool Description

Chapter III discussed a series of design guidelines and presented an eleven step program development process meant to help system designers create more user-friendly programs. Chapter IV then described the program development process as it was applied to software development in this thesis effort. The purpose of this section is to give a more detailed description of the CAD normalization tool itself. To accomplish this end, some of the more significant design issues faced during the project will first be presented. Then, a brief description will be given of how the normalization tool is operated.

Design Issues

Rather than outline each normalization tool module in detail, I feel that it is more informative to discuss the system software in terms of the major design decisions or issues that had to be dealt with during the effort. The following discussion consolidates several design issues into four categories: system memory limitations, CRT display, mouse implementation, and system file manipulation.

System Memory Limitations.

As mentioned in Chapter IV, it was anticipated that the internal memory capability of the LSI-11 could significantly influence the normalization tool design. In addition to the memory required for program code, a great deal of dynamic memory allocation is required to store the relational information manipulated by the tool software. Jankus, in his mouse prototype, had been forced to divide the software into two separate programs, one that handled the specification of functional dependencies, and another that accomplished the nonloss decomposition. It was expected that a similar approach would have to be taken in the re-design, due to the fact that modules were to be added to the existing software.

As modules were added to the program, it became necessary to lessen the amount of dynamic allocation by reducing the number of relations contained in test input files. For example, in early development of module DIS_RELATIONS which displays the names of a file's existing relations, it was possible to upload approximately twenty relations of seven attributes each. Using a file of this size it was possible to test and trouble shoot the modules that perform the paging of a large number of relation names within the display area of the primary CRT. As program software expanded, however, it was necessary to reduce dynamic allocation by uploading only two test relations. This reduction did not adversely affect program development, because new software could still be tested and validated with the smaller relation set.

Limited memory came to have more serious implications as graphics library modules were linked into the program. These modules were to be used to generate functional dependency diagrams as the user specified a relation's dependencies.

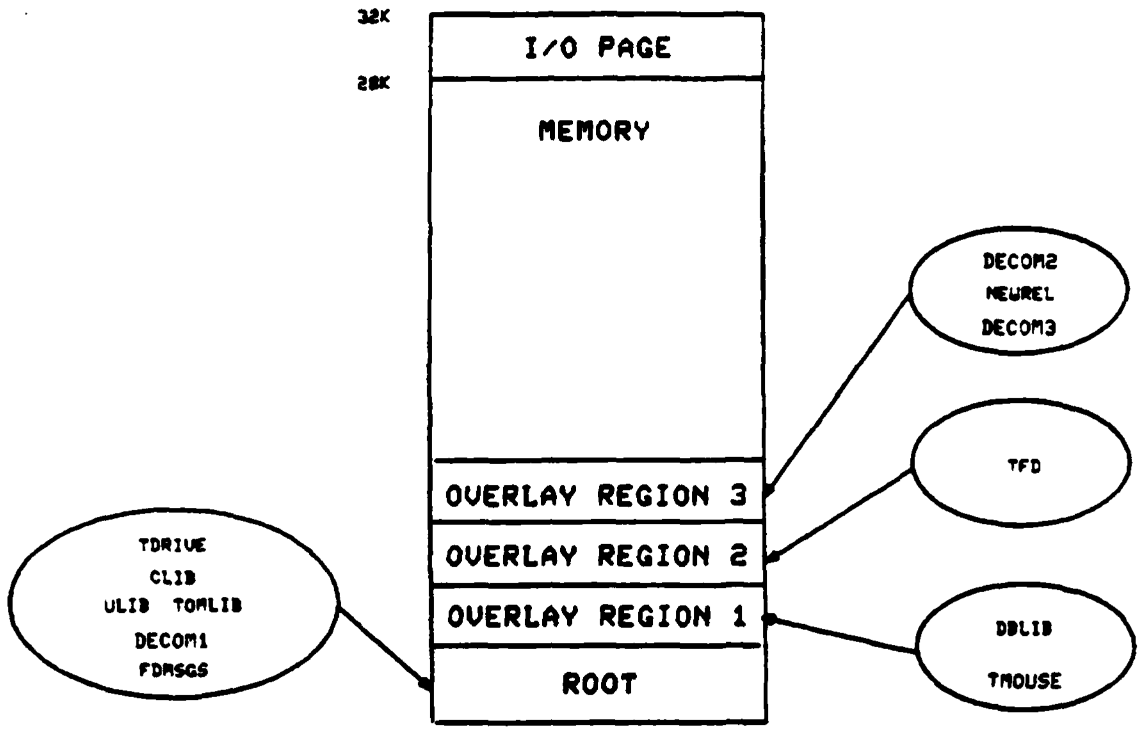
Due to the added size of the graphics modules, sufficient memory was not available to accomodate necessary dynamic allocation. Due to this limitation, various alternatives were considered which would allow the loading of necessary program code and a minimal set of relational data that would permit further software development.

One alternative that was investigated was the use of the RT-11 operating system's extended memory (XM) monitor, a version that allows access to additional storage space in upper memory. Although the monitor had never been used in any applications in the Digital Engineering Laboratory, it was hoped that the monitor could be brought up with a minimal amount of difficulty. It soon became apparent, however, after extensive experimentation by lab personnel and students, that it was not feasible to implement the monitor in the time frame of the thesis effort. At the time of this writing, several engineering students are continuing work with the XM monitor, in hopes that it can be used in future applications.

Another alternative considered was to move the normalization software to another computer within the laboratory, a system which had greater memory capability and also had the graphics hardware required by the graphics libraries. Although this approach would have solved the problem of available memory, new software would have had to have been developed to allow the use of several of the tool's peripheral devices, including the mouse and the secondary CRT display. Within the time constraints of the thesis effort, it was felt that this approach would not be workable.

Considering these other alternatives, it was finally decided to use the LSI-11's memory overlay capability. Implementation of the overlay method, which did not require the XM monitor, involved segmenting the software into independent pieces of code that could be brought into lower memory when required. Figure 4 (a) and (b) show the resulting memory map using the overlay scheme, and the link statement required to invoke the overlay technique. Even using this technique, it was not possible to link in the graphics library and still have sufficient memory space for dynamic allocation. However, a workable solution was found for this problem, as will be discussed in the following section on CRT display.

The problems associated with available memory were considerable, resulting in program design changes, but also led to greater understanding of how LSI-11 memory management is accomplished. Use of memory overlays allowed development of normalization tool software that was necessary to achieve overall system objectives. The amount of input data manipulated by the tool had to be reduced even with the overlay technique. However, the quantity of data was more than sufficient to test and validate specified tool features.



(a). LSI-11 Memory Map (27:4-4)

The following statements are contained in file FDTOOL.COM, and comprise the link statement used to implement the program's overlay scheme:

```
      R LINK
      FDTOOL,FDTOOL=CHDR/B:3000/M:3000//
      TDRIVE,DECOM1,ULIB,TOMLIB
      FDMSGS,CLIB
      DBLIB/O:1
      TMOUSE/O:1
      TFD/O:2
      DECOM2/O:3
      NEWREL/O:3
      DECOM3/O:3
      //
      ^C
```

(b). Program Link Statement

Figure 4. Program Overlay Scheme

CRT Display Method

A second project design category that merits further discussion is that of the techniques used for the display of information to the CRT. Information display, as addressed in Chapters III and IV, was a very important issue in the design of a user-friendly system. The discussion that follows deals with some of the problems encountered in designing the CRT displays.

Two Heathkit H-29 Video Display Terminals were used in the normalization tool. The primary terminal was configured with a Cleveland Cadonics Graphics Card, providing graphics features not available on the basic H-29. The card offered four different operational modes that could be used with the H-29 terminal: the H-29 standard terminal alphanumeric mode; the native graphics mode, used for generation of all graphics commands; the alphagraphics mode, used for entering text on graphic displays; and the tektronics mode which offered a wide range of graphics features. (For a detailed description of these modes refer to the Cleveland Cadonics User Manual (6).)

Since the majority of displayed information was to be text, a decision was made early in the development process to use the standard terminal mode for drawing the screen functional areas and for the display of alpha-numeric characters. This choice was made because the text displayed by the standard terminal mode had better definition and clarity than that of the alphagrahics mode. The terminal mode also had a set of graphic symbols that could be used to draw boundary lines on the screen, making the use of the native graphics mode unnecessary except for the drawing of functional dependency (FD) diagrams in the edit routine of the program. The intent was to use the standard terminal mode to display textual information to the menu, display, and message areas of the screen, then transition to the native/alphagraphics modes only for display of FD diagrams in the workspace area. Figure 5 shows a sample of the functional screen layout.

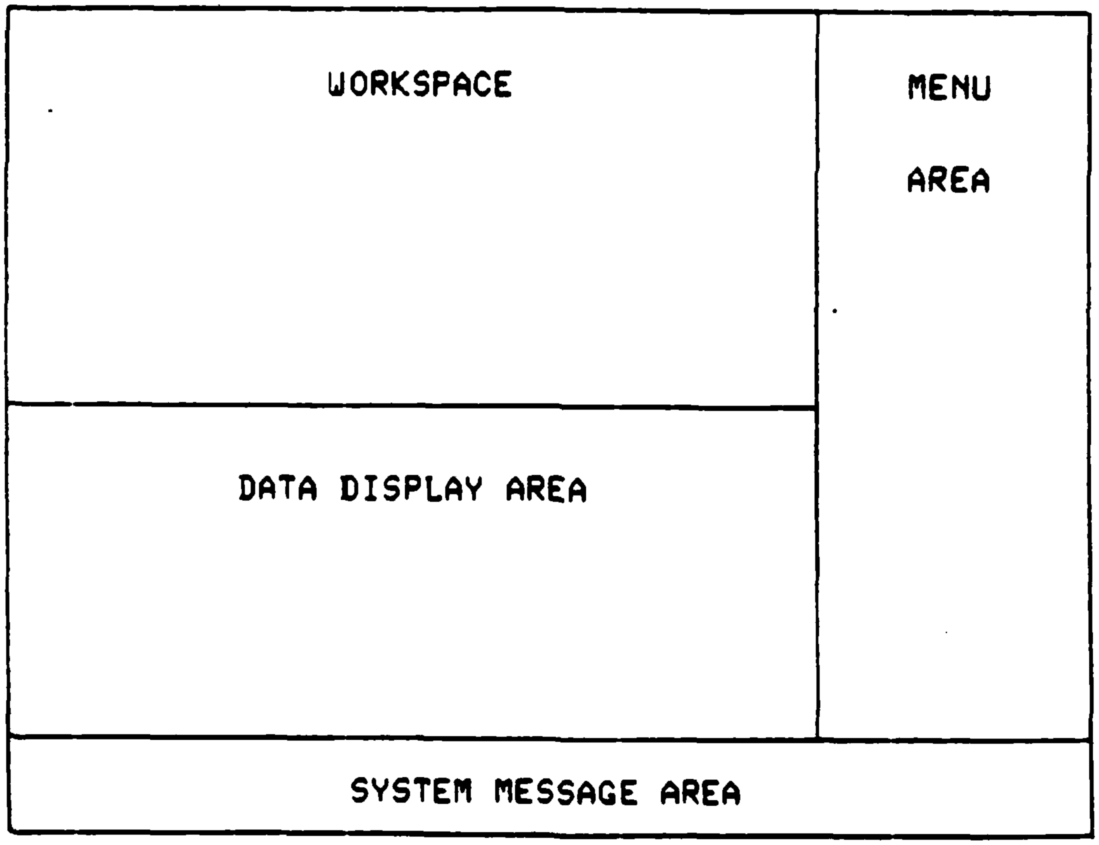


Figure 5. Sample Functional Screen Layout

The transition from one H-29 terminal mode to another was accomplished by sending a software escape sequence to the terminal. It was not discovered until mid-way through the effort, that a transition to either the native or alpha-graphics mode caused all existing information on the screen to be erased. This presented a problem because the user needed the textual menu and display information, generated in the standard terminal mode, at the same time the FD diagrams were being displayed in the workspace.

To resolve this problem, a modified version of code was created to generate the entire screen display, layout and text, in the native and alphagraphics mode. This proved to be a fairly trivial task due to the modular design of the program, and involved changing all program PRINTF calls to commands compatible with the Cleveland Cadonics instruction set. The resulting display was not as clear as that generated in the standard mode, and was characterized by a distracting visual flicker and a slower display of textual information. Even with these drawbacks, however, the display was usable and efforts turned to creating modules which drew the FD diagrams.

It was at this point that the memory limitations, discussed in the previous section, hampered further development of graphics displays using the native and alphanumeric modes. The addition of new diagram-drawing modules that used the graphics libraries, required so much additional space that the LSI-11 had insufficient memory to load even the program code. Attempts were made to use memory overlays similar to those used in the non-graphic version. However these attempts were unsuccessful.

As an alternative, test code was written to see if the functional dependency diagrams could be drawn strictly using the graphics set contained in the standard terminal mode. Such an approach was feasible because the set contained line segments that could be used to form the boxes and arrows required for the diagrams. Some flexibility would be lost in terms of drawing arrows at various angles; however this was not considered to be a serious drawback, because effective graphical representations could still be created using this pseudo graphics approach.

The elimination of the graphics libraries, together with the use of memory overlays for the remaining code, resulted in significant memory savings. So much space was saved, that it became possible to link the decomposition modules into the program. It was then possible to perform both the specification of functional dependencies and the non-loss decomposition of relations within one program, instead of separating the two routines as had been done in the Jankus prototype. Consolidating the two routines into one program contributed to making the tool easier for the DBA to use.

Mouse Implementation

A third major design issue raised during the project was that of developing software required to interface the SummaMouse as the primary data input device. Jankus had used the mouse as an attribute selection device in his prototype; thus he had developed code necessary for communication between the mouse and the host computer. To use the mouse as a selection device for more than one type of data, i.e. menu items, relation names and attribute names, it was necessary to develop new modules capable of differentiating between the different types of data selected from the CRT.

Jankus' modules TWEEK_MOUSE, INTERPRET_MOUSE and CALIBRATE were used to update horizontal (x) and vertical (y) coordinate information defining the mouse's position on its pad. These (x,y) coordinate values were then scaled in module MOWSE2SCREEN, converting the mouse pad coordinates to values corresponding to the 80 column by 25 line coordinate system of the display terminal. The terminal cursor position could then be updated to the (x,y) location on the screen. By interpreting the position of cursor at the time the mouse pushbutton was depressed, it was possible to determine the item at the cursor position that the user wanted to select.

Figure 6 shows the functional areas of the display terminal, together with the column and line numbers used to define each area. The menu area, for example, occupies columns 64 thru 80 and lines 1 thru 21. Similarly, the

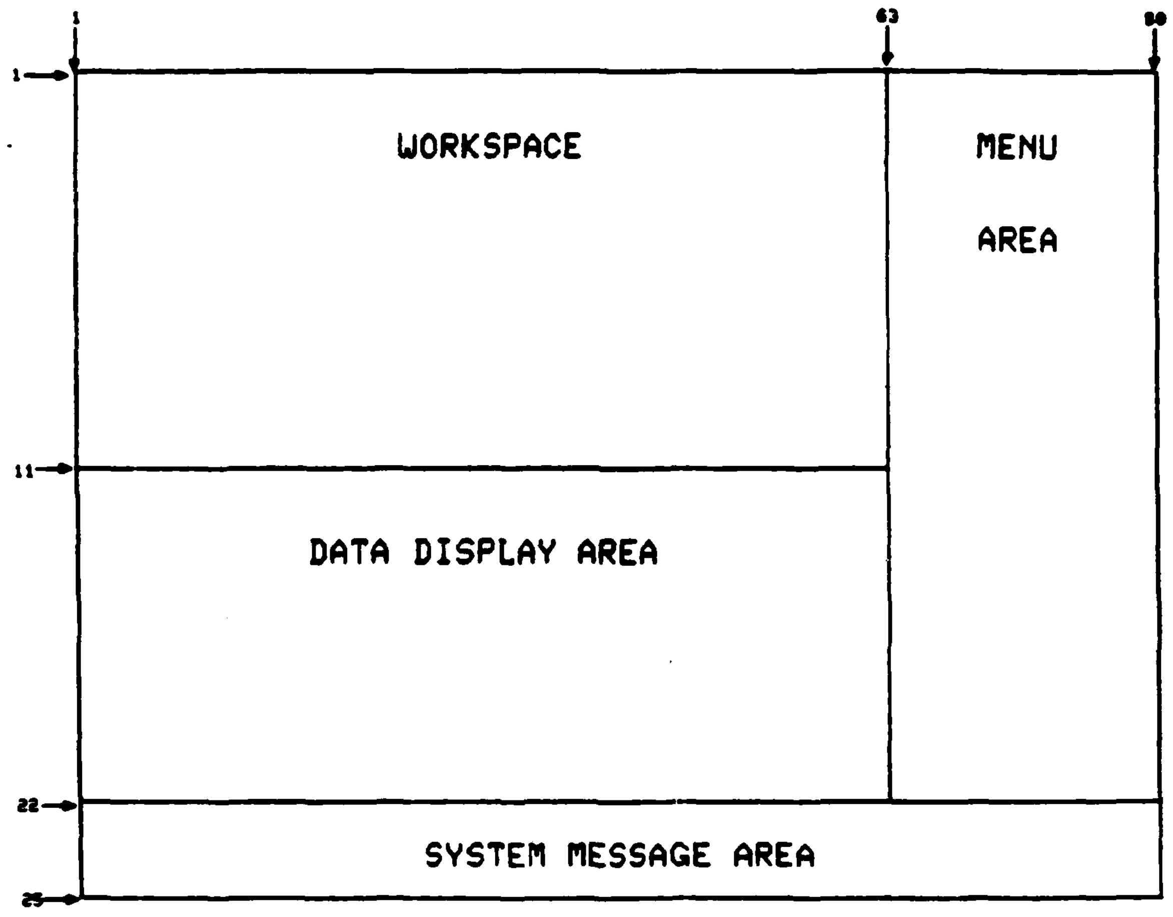


Figure 6. Screen Column and Line Format

message, display and workspace areas are bounded by specified column and line parameters within the system software. Although the display cursor appears to have free movement within all areas of the screen except the message area, modules FIND_SCREEN_LOCATION and MOWSE2SCREEN actually direct the cursor to one of four logical columns on the screen, columns 1, 21, 42 or 64 and line numbers that lie in the range of 1 to 21.

This four column scheme was used to simplify the task of selecting various items from the screen. For example, if the user has the option of either selecting a relation name from the display area or choosing a menu item, the fact that the selected column number is 64 indicates to the software that a menu choice has been made. Likewise if a returned column number is 1, 21, or 42, and the corresponding line number is between 12 and 21, the software recognizes that the choice must be a relation name. Having made this column determination, either module *GET_MENUCHOICE or SEEK_RELATION is then* called to determine the specific item selected within the functional area. In the case of a menu item, GET_MENUCHOICE uses the line number of the selected item to search a linked list data structure containing the names of the currently displayed menu, and identifies and returns the menu item selected. Module SEEK_RELATION uses a similar scheme to identify a specific relation name chosen from the display area.

In addition to developing software to handle the selection of data items, some minor modifications were made to Jankus' mouse calibration routines. The calibration process was streamlined, making it easier for the user to perform.

Also the problem, as discussed in Chapter II, of inverse mouse/cursor movement in the vertical direction, was corrected. These changes helped make the mouse easier to use and reduced some of the irritants cited by students who used the Jankus prototype.

System File Manipulation

A fourth design issue faced in the development of the normalization tool was the way in which the system manipulates data files. The Jankus prototype restricted the user to a default file name that contained a set of relations to be normalized. In the re-designed system, the user is given the opportunity to choose the file that he wants to work on, giving him greater flexibility. Likewise, on exiting the tool, the user can specify the name of the file in which the current set of relations is to be stored, rather than overwriting the file that contained the original relations. Therefore, if the user wants to maintain the original set of relations that existed prior to the normalization session, he can do so simply by selecting a new name for the output file. Using this scheme, the DBA can set up his own archive system, keeping a record of all relations prior to any normalization.

The system also automatically creates files containing any partially specified dependencies created during a normalization session. If the user is working on a large relational set, and he wishes to exit and complete the normalization process at a later time, all previously specified dependencies for an unnormalized relation are automatically saved. The system appends a ".SVE" extension to the user-selected output filename, and stores the dependencies for future use. When the DBA later uses the tool and enters a filename, the system searches for that filename with a ".SVE" extension. If the file exists, the dependency data is uploaded for use in the current session. This file manipulation remains transparent to the user, so all the DBA has to worry about is the name of his desired input file, and the file in which he wishes to store his output information.

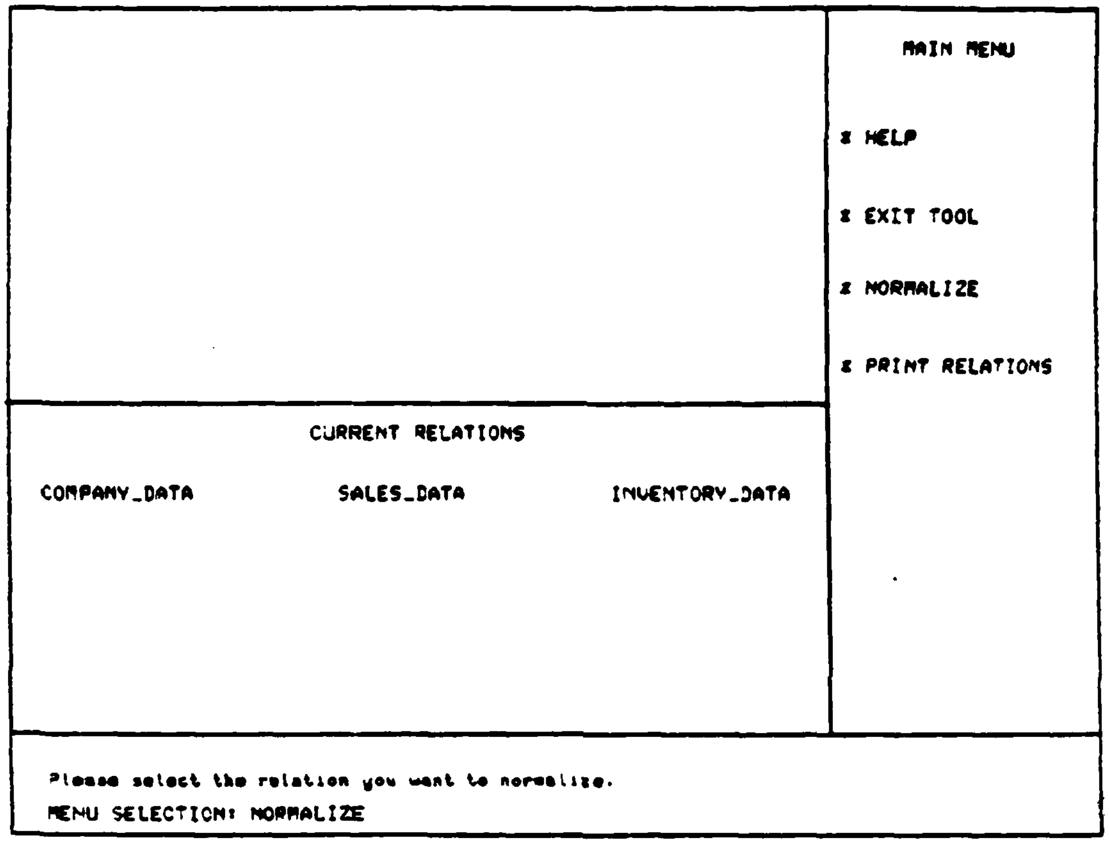
The normalization tool also uses file storage for all online help information displayed to the user. When the user selects a help menu item, the system software seeks and displays the appropriate help text file on the secondary CRT. In light of the memory limitations of the host system, this file scheme was selected so that the large amount of textual information would not have to be kept in internal memory. This method also allows changes to be made to the displayed help information, without having to search through the tool's source code. Similarly, all user prompts and system messages are maintained in a separate file, simplifying the task of making changes to the displayed information.

Normalization Tool Operation

Having looked at some of the more significant design aspects of the normalization tool, this section gives a brief description of how the tool operates. The discussion will be in rather general terms, with a more detailed description contained in the system user's guide found in Appendix A.

Before beginning a session, the DBA must ensure that the system's executable code (file FDTOOL.SAV), the desired input file, and the system help files are contained on the host computer's default drive. To call up the tool, the command "RUN FDTOOL", is typed. A brief introduction is displayed, along with a prompt for the user to enter the name of the desired input data file. Once a valid filename is entered, the user is then asked if he would like any help information concerning general tool operation and is instructed to calibrate the mouse if it has not previously been powered up. When calibration is complete, the main menu level (Figure 7) is displayed on the screen, together with the names of the relations contained in the input file. Once at the main menu level and all subsequent levels, the mouse is used to make all desired selections. The DBA simply uses the mouse to move the screen cursor over the desired item, and depresses the mouse center key to make a selection.

Four menu items, HELP, EXIT, MORE\/ and MORE/\ are common to all menu levels. When HELP is selected, on line help information is displayed to the secondary CRT, offering assistance on all current menu options at the given level. When the user wishes to continue, he depresses any mouse key, restoring any previous data on the secondary CRT and is allowed to make another menu choice. The EXIT menu option



Please select the relation you want to normalize.
MENU SELECTION: NORMALIZE

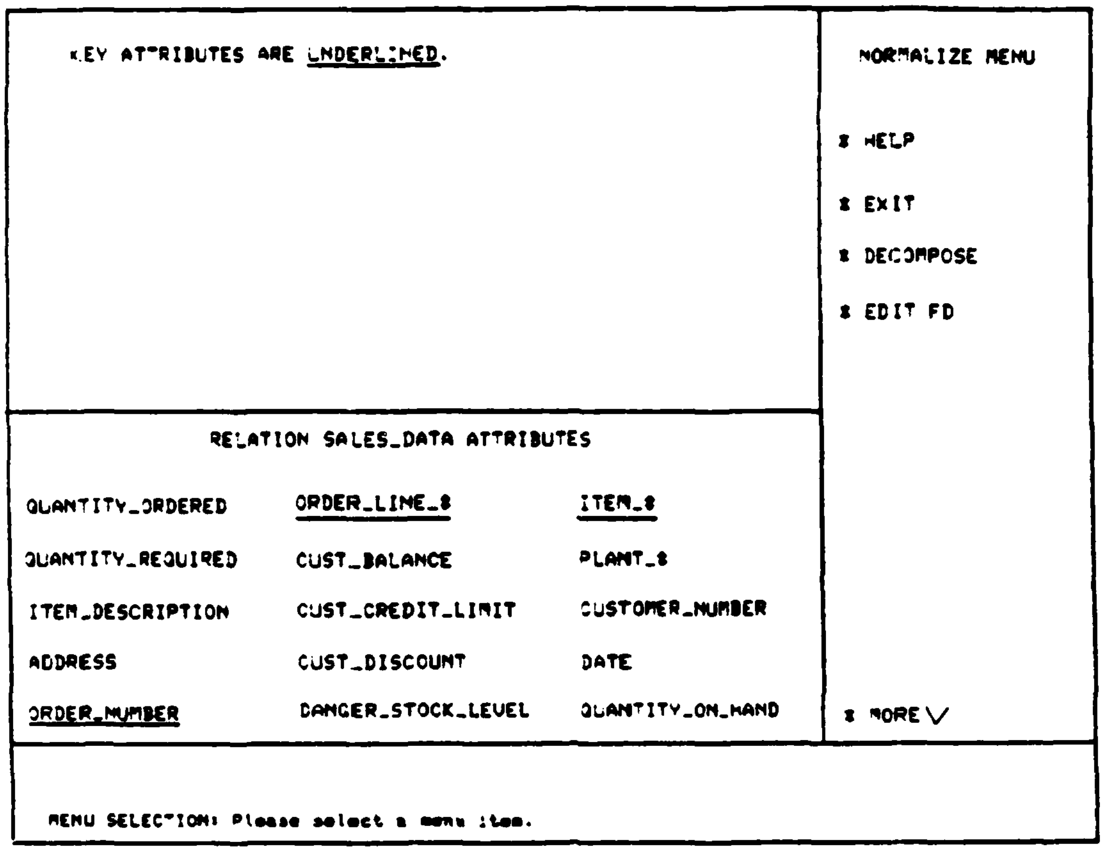
Figure 7. Main Menu Level

always allows the user to return to the next highest menu level. In the case of EXIT TOOL, file cleanup operations are performed, and the user returns to the operating system level. MORE\/ and MORE/\ are options that will appear in the menu area only if more relation or attribute items exist than can be shown in the display area of the screen. These options give the user a paging capability to view all pieces of data regardless of quantity.

Menu option PRINT RELATIONS gives the user the option of printing out in hard copy, any or all current relations and their associated attributes. This option gives the DBA an opportunity to print out a file's relations either before or after the normalization process is performed, putting the information in a text format that can be used for record keeping purposes.

Selection of the normalize option results in a prompt which asks the user to select the relation he would like to normalize. After receipt of a selection, the tool then transitions to the normalize menu level, where a new menu is displayed along with a listing of the attributes that comprise the selected relation (See Figure 8). If there are existing functional dependencies from a previous session (i.e. contents of a ".SVE" file), these dependencies are displayed on the secondary CRT.

Once at the normalize menu level, in addition to HELP and EXIT, the two options are EDIT FD and DECOMPOSE. The DBA selects the EDIT FD option when he wishes to specify any new functional dependencies or edit any dependencies that already exist. The EDIT FD menu level, shown in Figure 9, consists of eight possible items plus the two conditional paging options. All editing functions, select (SEL), delete (DEL) and save (SAVE FD) are performed on the functional dependency currently displayed in the workspace area of the primary CRT.



KEY ATTRIBUTES ARE UNDERLINED.

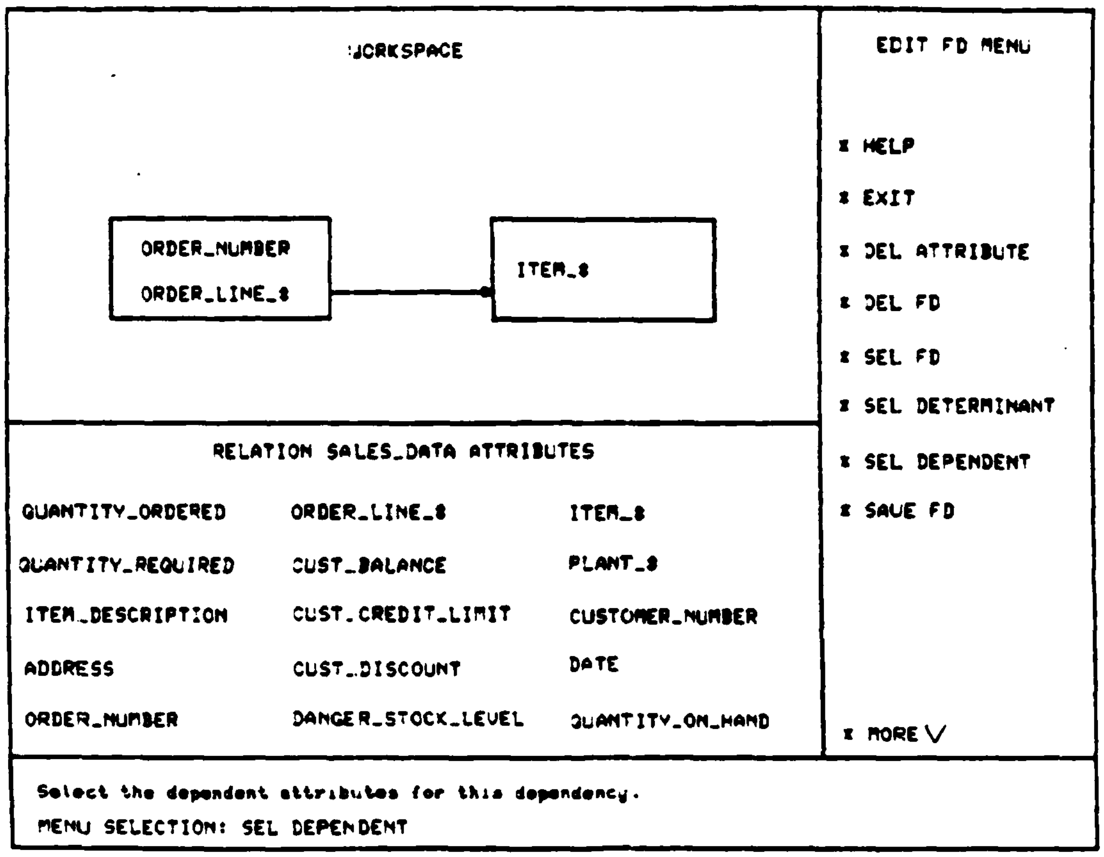
RELATION SALES_DATA ATTRIBUTES

QUANTITY_ORDERED	ORDER_LINE_#	ITEM_#
QUANTITY_REQUIRED	CUST_BALANCE	PLANT_#
ITEM_DESCRIPTION	CUST_CREDIT_LIMIT	CUSTOMER_NUMBER
ADDRESS	CUST_DISCOUNT	DATE
ORDER_NUMBER	DANGER_STOCK_LEVEL	QUANTITY_ON_HAND

MENU SELECTION: Please select a menu item.

Figure 8. Normalize Menu Level

For example, SEL DETERMINANT and SEL DEPENDENT are used to select and display a given attribute in the functional dependency diagram drawn in the workspace. SEL FD is used to select a previously specified dependency, displayed on the secondary CRT, and bring it into the workspace for further editing. Likewise, the delete options are used to delete either an attribute or the entire dependency from the



Select the dependent attributes for this dependency.
MENU SELECTION: SEL DEPENDENT

Figure 9. EDIT FD Menu Level

workspace. Selection of the SAVE FD option adds the dependency shown in the workspace to those existing dependencies displayed on the secondary CRT. At any time in the editing process, the DBA can exit to the normalize menu, where he can decompose the relation or return to the main menu level.

The normalize menu's DECOMPOSE option is selected when the DBA is satisfied that all functional dependencies have been specified, and he wishes to execute the nonloss decomposition of the current relation. Prior to the decomposition, the DBA may review the existing dependencies displayed on the secondary CRT, and the tool performs a check of the existing dependencies, to ensure that all attributes are included in at least one of the specified dependencies. If any attributes have been excluded, the tool warns the user, to prevent an attribute from being unintentionally deleted from the data set.

Upon completion of the decomposition process, the user is asked to select names for the resulting set of relations. The attributes that comprise each normalized relation are shown in the display area. The tool prompts, reads and error checks each new name entered by the user and returns to the main menu level when all relations have been been named. Once back at the main level, the names of the new relations and all other relations contained in the input file are displayed, and the DBA is free to normalize a new relation or exit as desired.

To provide consistency at all menu levels, line 24 of the primary display is used to show the current menu item selected, or else to inform the user that the system is waiting for a selection. Line 23 is also used to provide feedback to the user. This feedback is in the form of either short prompting messages, or error messages if the user makes an inappropriate selection. Visual attributes such as underlining and reverse video are also used throughout the program to help add clarity to the display. For example, all prompting and error messages on line 23 are shown in reverse video to draw the user's attention to the displayed information. These and other techniques were used in an effort to continually inform the user of where he is in the program, and to make him aware of his possible options.

The preceding description of the CAD normalization tool was meant to provide some insight into some of the major design issues faced in developing the tool, and to give general information on how the tool is operated. The next chapter will describe the results of an evaluation conducted by a group of AFIT students who had an opportunity to use the normalization tool.

VI. CAD Tool Evaluation

As discussed in Chapter III, the term user-friendly is not one which lends itself to a simple, commonly accepted definition among either design experts or among potential computer system users. It follows that the task of measuring the extent of a system's user-friendliness is equally challenging, or even more difficult than trying to define the term itself. This fact soon became apparent during the evaluation stage of this project. In conducting the system evaluation, the intent was to receive feedback from potential users on the system's user-friendliness, and on the degree of satisfaction stemming from use of the design tool.

The remainder of this chapter describes the evaluation of the normalization tool by a group AFIT graduate students. First, the tool used to measure and analyze computer user satisfaction will be presented. Then, the methodology of the CAD tool evaluation will be described. This discussion includes a description of the evaluation test group, as well as a description of the test scenario and test materials made available to the group. Finally, the results of the evaluation will be presented.

Measurement Tool

Measuring user satisfaction of a computer system is an area of study that is of high interest to both behavioral scientists and system designers alike. Powers and Dickson conclude that user satisfaction is the most critical criterion in measuring computer system success and failure (22:530). One obvious method of judging the extent of satisfaction, is to observe the extent to which people are willing to use a newly designed system. By using interviews, questionaires or simply tracking the amount of time the users spend on a system, managers can get an intuitive feel for user attitudes toward a new system. However, it would be advantageous to have a more quantifiable measure of satisfaction that could be used prior to actual system implementation. There is very little information found in current literature that indicates the existence of proven tools to measure and analyze either user-friendliness, or system user satisfaction.

Bailey and Pearson, however, present a technique to measure and analyze user satisfaction of information systems, a technique that would appear to be adaptable to other types of computer systems as well. Based on current behavioral science research, Bailey and Pearson contend that user satisfaction is the sum of one's positive and negative reactions to a set of factors relative to a given situation. Satisfaction can then be defined as the sum of the user's weighted reactions to a set of factors,

$$S_i = \sum_{j=1}^{n} R_{ij} \quad W_{ij}$$

where, R_{ij} = the reaction to factor j by individual i, and W_{ij} = the importance of factor j to individual i (22:531).

Using this definition, it is necessary to determine the relevant factors (R), as well to develop a vehicle for scaling the individual's reaction to those factors.

To determine information system user satisfaction, Bailey and Pearson developed a list of 39 factors that various experts agreed were of concern to system users. These factors covered a broad spectrum, ranging from the user's perception of his relationship with the EDP staff, to the user's feeling of confidence in the information system output (22:539-542). Using a semantic differential technique, four unique bipolar adjective pairs were devised to describe the characteristics of each of the 39 factors. In addition, two common adjective pairs were used for each of the factors: satisfactory - unsatisfactory, used to measure the internal consistency of the four unique adjective pairs; and the word pair important - unimportant, used to measure the weight given to the factor (W in the given equation). An example of one of the factors as it appears in a questionaire form is shown in Figure 10. As shown in the sample, a seven interval scale is used for each word pair, in which the user can express his feelings relative to a given factor.

The scaling of the seven intervals was quantified by assigning the values -3, -2 .. +2, +3 to the intervals. The importance scale was assigned values from .10 to 1.00 in increments of 0.15. Using these values, the user's reaction to a given factor can be computed as the average response to

23. LANGUAGE: The set of vocabulary, syntax, and grammatical rules used to interact with the computer system.

simple								complex
powerful								weak
easy								difficult
easy-to-use								hard-to-use
satisfactory								unsatisfactory

To me, this factor is:

important								unimportant

Figure 10. Sample Questionaire Form

the four adjective pairs:

$$R_{ij} = 1/4 \sum_{k=1}^{4} I_{ijk} \quad \text{where}$$

I_{ijk} = the numeric response of user i to adjective pair k of factor j.

Summing the individual weighted factor responses over all 39 factors, the overall user satisfaction becomes:

$$S_i = \sum_{j=1}^{39} W_{ij}/4 \sum_{k=1}^{4} I_{ijk}$$

The range of S is from +117 to -117. Bailey and Pearson recommend a normalization of the resulting score to a range from -1 to +1. (See reference (22) for details.) The normalized scores can then be translated to yield a measure of user satisfaction as shown in Figure 11.

NORMALIZED SCORE	TRANSLATION
+ 1.00	Maximally satisfied
+ 0.67	Quite satisfied
+ 0.33	Slightly satisfied
+ 0.0	Neutral
- 0.33	Slightly dissatisfied
- 0.67	Quite dissatisfied
- 1.00	Maximally dissatisfied

Figure 11. Score Boundaries For
Normalized User Satisfaction (22:535)

Evaluation Methodology

The measurement and analysis techniques described in the previous section were adapted to gauge the degree of user satisfaction in the normalization tool developed in this project. Since the effort's main area of interest was the system's degree of user-friendliness, a questionaire was created, consisting of eight factors related to the human-computer interface. Using a semantic differential technique, four adjective pairs were selected for each factor, along with the importance and satisfaction word-pairs used by Bailey and Pearson. The complete questionaire used in the evaluation is shown in Appendix B.

The majority of test subjects used in the tool evaluation were graduate students enrolled in AFIT EENG 793, Advanced Software Engineering. It was felt that the students' expertise in the areas of software design and development would be helpful in providing objective feedback on the interface of the normalization tool. The use of this group, however, did induce a degree of artificiality in the evaluation. The target user for the normalization tool was specified as an experienced DBA or a member of his staff, individuals well-versed in the normalization process. Although several of the test subjects were familiar with relational database concepts, most did not possess as in-depth a knowledge of the normalization process as would be expected of the target user. Therefore, some individuals experienced some minor problems in using the tool, problems stemming more from task unfamiliarity, than from CAD tool deficiencies.

Task unfamiliarity was anticipated when designing the test scenarios used in the tool evaluation. Prior to using the tool, all students were given a brief overview of the normalization process, and of the concept of functional dependencies. In addition, rather than requiring each student to determine the functional dependencies relevant to a given attribute set, each student was given the real world dependency information. (See Appendix C for sample evaluation materials made available to the students.) Therefore in using the tool, unlike a DBA who would draw upon his knowledge of the relation information to specify dependencies, the students were asked to simply use the tool to enter information that was already given them. Although somewhat artificial, it was felt that this aspect would not seriously impact the results of the evaluation. The main objective was to receive feedback on the user-friendliness of the software interface, feedback that was successfully obtained.

A total of sixteen students and two faculty members were asked to participate in the exercise, with fifteen individuals actually completing the evaluation questionaires. Each individual was asked to use <u>two</u> different versions of the CAD tool in normalizing a given relation; version one was the original tool and version two the re-designed system. Both versions were evaluated, not so much to determine which was the better of the two, but rather to gain an insight into the interface features that potential users found to be useful for the given application. Since each test subject was asked to use both versions, half of the group was asked to use version one first, and the other half was requested to first perform the task using version two. This was done in an effort to detect any possible influence that exposure to one system might have on the evaluation of the subsequent system.

As shown in Appendix C, enough LSI/RT-11 system information was given to the individuals to get them to a point where they could invoke either of the two normalization tools. From that point, they were expected to perform the normalization task on the given relation information using the facilities of the respective CAD tool. Copies of the user's guide for each version were also made available during the normalization session and individuals were free to use this information as desired. After completing the normalization task using a given tool version, subjects were asked to complete a questionaire for that version before using and evaluating a second CAD tool. In addition to completing the formatted information on the questionaires, all individuals were encouraged to include any additional comments about the software interface.

Evaluation Results

Analysis of the evaluation data involved more than merely manipulating the empirical data obtained from the questionaires. Prior to accomplishing any comparative analysis on the degree of user-satisfaction provided by each of the two CAD tools, it was first necessary to examine the reliability and validity of the questionaire. Measurement tool validation was a necessity to gauge whether the tool measured what it was intended to measure. The discussion that follows first addresses three categories of validity relevant to the measurement tool: content, predictive, and construct. Then, the comparative analysis performed on questionaire results will be described in detail.

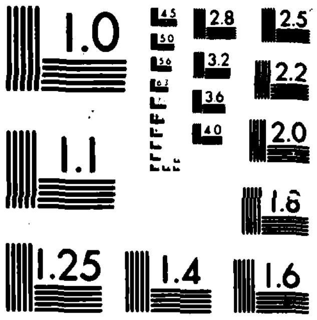
Questionaire Validation. The user responses to version two of the CAD tool were used to measure the validity of the questionaire. The first category examined was content validity. Content validity implies that all aspects of the attribute or factor being measured are considered by the tool

```
AD-A164 100   DESIGN OF THE HUMAN-COMPUTER INTERFACE FOR A COMPUTER      2/2
              AIDED DESIGN TOOL F.. (U) AIR FORCE INST OF TECH
              WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..  T C MALLARY
UNCLASSIFIED  DEC 85 AFIT/GCS/ENG/85D-8               F/G 9/2       NL
```

END

FILMED

DTIC


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

(22:536). The attainment of content validity is to a large extent a subjective task that may depend on a trial and error process. Nunnally alludes to this fact when he says there are two major standards for ensuring content validity: a representative collection of items, and "sensible" methods of test construction. (21:92).

Pearson and Bailey offer a possible measure of content validity when they suggest a test of the correlation among the adjective pairs for each of the factors of the user-satisfaction questionaire (22:536). The rationale is that scales which are designed to measure the same factor should be positively correlated. Correlation testing was performed on the four adjective pairs for each of the eight factors used in the this project's questionaire. The results of the testing, shown in Figure 12, were obtained by correlating the user responses for each adjective pair, with the sum of the responses for the other three word pairs for a given factor. (See Nunnally (21) for a more detailed description of the methodology). A T distribution was used to test the null hypothesis that the correlation values were positive. The asterisked word pairs in Figure 12 indicate those pairs whose correlation were not significant at a .10 level. These results would seem to indicate the trial and error nature of designing such a measuring device, and provide an indication of those elements of the questionaire that could be improved.

Predictive validity implies that a measuring tool is

ADJECTIVE PAIR	MEASURED CORRELATION			
	FACTOR 1	FACTOR 2	FACTOR 3	FACTOR 4
1 vs. 2,3,4	.3891553	.3583003	.5298447	.454714
2 vs. 3,4,1	.5378695	*.1524755	.5298447	.440426
3 vs. 4,1,2	.6069636	*.2341877	*.1233598	.4197465
4 vs. 1,2,3	.660533	*.2260659	.472153	.5412017
	FACTOR 5	FACTOR 6	FACTOR 7	FACTOR 8
1 vs. 2,3,4	.547549	.444061	.577177	.586252
2 vs. 3,4,1	.5750755	*.3291744	.445097	.490127
3 vs. 4,1,2	.5750755	*.2634103	*.3047183	.455589
4 vs. 1,2,3	.478115	*-.2486472	.472539	.596143

Figure 12. Adjective Pair Correlation Results

consistent and agrees with other possible measures. To gauge the predictive validity of the questionaire, each individual's response to a factor's unsatisfactory-satisfactory word pair was correlated with the computed satisfaction index for a given factor. These correlation results are shown in Figure 13. Once again, a T distribution was used to test the hypothesis that positive correlation existed between the measured level of satisfaction and the respondent's self assessment of satisfaction. All eight correlation coefficients were found to be significant at a 0.01 level.

A third validation category, construct validity, relates to the extent that a va.iable of interest is abstract rather that concrete (21:96), and generally implies that a measuring

FACTOR	MEASURED CORRELATION
1	.72755
2	.69415
3	.638271
4	.798461
5	.89795
6	.870959
7	.78324
8	.842339

Figure 13. Measured Versus Self-Assessed Satisfaction Correlation Results

instrument performs as expected, relative to the abstraction of the factor being measured (22:536). It was not clear what, if any, empirical measures could be used to guage the construct validity of the project questionaire, so this issue was not addressed in any detail. (For more information concerning this topic, see Nunnally (21).)

A final questionaire validation issue related to the users' responses to each factor's importance word-pair. As indicated earlier, the eight factors included in the questionaire were chosen based on an intuitive feel of features pertinent to the human computer interface. Questionaire results were analyzed to determine if the users also felt that the factors were important. Taking the average of all fifteen respondent's importance ratings, scores for the eight factors ranged from 0.78 to 0.87 on a scale from zero to one. Factor 3 (Program Pacing) and factor 7 (Feeling of Control) received the lowest average importance ratings of 0.78 and 0.81 respectively. Likewise, factor 4 (Error Recover) and factor 6 (Ease of Learning) received the highest average ratings of 0.87 and 0.88. The fact that all eight factors received ratings on the upper-end of the unimportant-important scale can be interpreted to mean that the respondents did consider the factors to be important. However, a more detailed methodology for designing the questionaire, as used by Bailey and Pearson (22:531-533), would certainly be recommended in order to gain further confidence in the questionaire's measurement accuracy.

Analysis. Having addressed the validity of the project's questionaire, attention turned to a comparative analysis of the questionaire's data. Initial observation indicated that version two of the CAD tool received a higher user-satisfaction rating than did version one. The normalized user-satisfaction ratings for each version are shown in Figure 14. With the possible rating values ranging from -1.0 (maximally dissatisfied) to +1.0 (maximally satisfied), the results indicate that all respondents gave higher satisfaction ratings to version two than were given version one. Further analysis was conducted to test the statistical significance of the findings.

Analysis involved testing the null hypothesis that there was no significant difference between the version one and

Rater #	Version 1 Rating	Version 2 Rating
1.	.32	.56
2.	.07	.24
3.	.32	.58
4.	-.19	.93
5.	.28	.55
6.	-.05	.63
7.	.28	.57
8.	.35	.91
9.	-.32	.50
10.	.19	.47
11.	.46	.96
12.	-.51	.70
13.	-.19	.22
14.	.15	.75
15.	-.007	.43
	Mean = .077	Mean = .60

Figure 14. Questionaire Satisfaction Index Summary

version two results, versus the alternate hypothesis that version two received a higher rating of user-satisfaction than did version one. Due to the small sample size, a T Distribution was selected to test the hypothesis. In using the T Test, the assumption was made that the sample ratings for each version came from a normal distribution. A second assumption that <u>should</u> be made when using a simple T Test is that the individual observations from each sample are independent of one another. As mentioned earlier, the independence of the two ratings was questioned at the outset, because both versions were evaluated by the same group of people. Recognizing the possibility that the ratings of one tool could be biased by exposure to another, two different analysis techniques were used to reduce the impact of rating dependence.

The first technique used was a paired T test. Using this method, the measured variable was the difference between the two ratings for each of the fifteen test subjects. It was assumed that the differences (D-values) were normally distributed, and independent of one another. The paired T value was computed using the equation

$$T = (\bar{D} - \Delta_0) / (S_D / \sqrt{n})$$

Using the computed values, $\bar{D} = .823$, $S_D = 1.19$ and $n = 15$, the resulting T value was 2.678. Using a .01 level of significance, with 14 d.f., the T statistic used was 2.624. Since the computed T value was greater than the T statistic, the null hypothesis was rejected, lending support to the contention that the ratings given version two were higher than those given version one.

As an alternative analysis technique, the evaluation results were divided into two groups based on which tool a given test subject used first. Since ten individuals rated version one first, and five used version two, the sample sizes used were reduced in size to 10 for version one and 5 for version two. The two samples were again assumed to be normally distributed and independent. An F test was used to verify the necessary assumption that the sample variances were equal. A two-sample T test was computed using an alpha value of .01 and 13 d.f., yielding a T statistic of 2.650. Using the sample means shown in Figure 15, and a pooled estimator of the common variance equal to .081, the computed T value was equal to 10.219. Since the computed T value was much greater than the T statistic, the null hypothesis was again rejected in favor of the alternate hypothesis that the level of user-satisfaction for version two was significantly higher than that for version one.

In addition to providing a potential empirical measure of relative user-satisfaction, the questionaire's comment sections gave the test subjects an opportunity to express individual likes and dislikes pertaining to each of the normalization tools. Because the version one comments were

VERSION 1		VERSION 2	
Rater #	Rating	Rater #	Rating
1.	.32	5.	.55
2.	.07	6.	.63
3.	.32	7.	.57
4.	-.19	10.	.47
8.	.35	13.	.22
9.	-.32		
11.	.46		Mean = .488
12.	-.51		
14.	.51		
15.	-.007		
	Mean = .0343		

Figure 15. Alternate Data Analysis

nearly identical to those discussed in Chapter II, they will not be addressed here. The following is a brief discussion of the most frequently critiqued interface issues pertinent to version two.

The majority of the respondents commented that the menu driven version was a noticable improvement over the original tool, lending support to the empirical findings just presented. Several favorable comments were received concerning the system feedback, ease of use and help features of the tool. Although several individuals responded that mouse operation was easier in the modified CAD tool, the use of the mouse in general, was the item most frequently criticized by test participants. Criticisms varied from total dislike of the mouse in any application of this type, to minor irritation with mouse manipulation on its pad. Several people also commented on the placement of the data paging items (MORE\/, MORE/\ ) within the menu area. The point was made that the function of the items may have been more apparent had they been placed within the data display area of the screen. Two individuals also felt that the primary display was at times <u>too</u> busy with information. These, and other comments provided useful system feedback, feedback that can help a system designer in his efforts to create a user-friendly system.

VII. Conclusion and Recommendations

The purpose of this thesis effort was to re-design an existing computer aided design tool used to normalize the relations of a relational database. Primary emphasis was placed on researching state of the art techniques in human computer interfacing, and incorporating human factor concepts in the design of a more user-friendly system.

An appreciation was gained for the complexity of defining or describing what is meant by a user-friendly system. All too often, system users cannot even describe a friendly system until they see it, complicating the task of the system designer. Due to this fact, the software designer must have an awareness of certain design guidelines that can help him in the program design and development process.

Material discovered in the literature search, as well as practical experience gained in the design of the project's software, reenforced the fact that user-friendliness must be an integral part of the design process, and not something that can be added as an after-thought. The application of a few sound software engineering principles throughout the development process can go a long way in producing a more user-friendly system. The program development process presented in this report, although similar to other popular approaches, places greater emphasis up front on the design of the human-computer interface.

In developing the CAD tool software of this project, it was clear that certain design trade-offs may have to be made in creating a friendly interface. For example, the memory and graphics limitations encountered during the effort resulted in adjustments to the original interface design. The hardware limitations to some extent resulted in a re-thinking of the desired interface. Judging from material found in current literature, trade-offs of this type are not uncommon when designing interactive systems.

Finally, this project addressed the difficulties associated with trying to quantify the extent of user-friendliness, or in measuring the degree to which a user is satisfied with a computer system. The questionaire adapted to serve as a measurement tool in this project, should be viewed as a starting point from which other, more precise measuring devices may evolve.

As a result of this thesis effort and the previous work by Jankus, there are several areas recommended for further study. The normalization tool is currently limited to a reduced input data set because of the memory limitations of the LSI-11 and its RT-11 operating system. Work needs to be done in resolving these memory problems. As an alternative, with the expected arrival of the Digital Engineering Laboratory's new software work stations, the tool could be moved to a new system with greater memory and graphic capabilities.

With the critiques received concerning use of a mouse as the CAD tool's selection device, more work could be done in evaluating and implementing alternative input devices. New devices such as a light pen, a speech recognition system, and an improved mouse are now either available or on order for the DEL, making it possible to investigate the use of these devices as part of the human-computer interface.

Another area that could be examined is extension of the tool to perform the normalization process to fifth normal form. As Jankus points out, there is very little published information detailing an algorithm for this task, making it a challenging thesis for someone interested in database operations (16:72).

A final area of interest would be to investigate the development of better measurement tools so that designers can evaluate the extent of computer user-satisfaction. The questionaire used in this project was essentially a first-cut at such a tool. Using material available in behavioral science literature, additional research would be useful in identifying and validating those factors which can be used to measure user-friendliness and computer user-satisfaction.

APPENDIX A

User's Guide To The Computer Aided Design Normalization Tool

(1.0) Introduction

This document is designed to supplement on-line help facilities of the Relational Database CAD Normalization Tool. First, an overview of the system functions is given, describing how the tool can assist the Data Base Administrator. Next, general system features are described, features that include system setup, file manipulation and system interface. Finally, a step by step guide to system use is given, together with a description of the various menu levels of the tool.

(2.0) System Overview

The Normalization Tool has two primary functions:

(1) Provide the DBA with an interactive method of specifying functional dependencies that exist among a set of relational attributes.

(2) Use the specified dependencies to perform a nonloss decomposition, creating new relations that are normalized to third normal form.

The CAD tool is designed to help the DBA normalize a set of relations used by the Digital Engineering Laboratory (DEL) Roth Relational Database Management System. The tool manipulates data files created by the DBMS utility program Data Base Manager (DBMGR), and performs a nonloss decompostion of the relations, based on a set of user-specified functional dependencies. The new relations are subsequently output in a file format that is compatible for use by the DBMS. It should be noted that the system, as well as the material that follows, has been designed from the perspective that the intended user is a DBA or member of his staff. That is, it is assumed that the intended user has a basic understanding of a relational database, functional dependencies, normal forms, and has a working knowledge of the computer system that hosts the tool.

(3.0) System Setup

The current version of the normaliazation tool is implemented on the DEL's System "A", an LSI 11/73. System A, using the RT-11 version 5.1 operating system, is configured with a mouse, used as the primary input device, and two CRT display screens. The system has a hard disk drive consisting of six logical devices, RK0: thru RK4:, in addition to two eight inch floppy disk drives, devices DY0: (lefthand drive) and DY1 (righthand drive).

(3.1) To power up System A, set the red master control switch, located on the system control box, to the ON position. Ensure that the box's switch labeled AUX 1 is also on, so that power is provided to the mouse. Once powered up, the system will automatically boot, resulting in a "." system prompt. After system boot, RK0: is the default device. To simplify subsequent file manipulation, use the system command

ASSIGN to assign the system default drive to the logical device that contains the program and data files used by the normalization tool. For example, if the required files are located on floppy disk in the lefthand disk drive, the command "ASSIGN DY0: DK:" will set the system default to DY0:. Likewise if the files are located on hard sector RK4:, the command "ASSIGN RK4: DK:" would be used. This procedure is recommended to preclude having to specify the logical device names during subsequent file operations.

(4.0) File Manipulation

The files that are needed to run the normalization include:

-- FDTOOL.SAV	the system executable code for the tool.
-- MENU.TXT	the system menus
-- HELP0.TXT	
-- HELP1.TXT	system text files containing help
-- HELP2.TXT	information for the various program levels.
-- HELP3.TXT	
-- HELP4.TXT	
-- <DATA.DAT>	a user specified file that contains the relation information to be normalized.

The system files must be present on the default system device to ensure proper system operation. In addition to these system files, you must ensure that the relation data file to be used by the tool is also available to the system. The data file, previously created using program DBMGR, contains all domain and attribute information for the relations to be normalized. If the file is not located on the assigned default device (RK0: .. RK5, DY0:, DY1:), any references to the filename at session start or finish must be prefaced by an appropriate device name. For example, if the default is RK0: and the input file INPUT.DAT is located on drive DY0:, the file would be referenced by DY0:INPUT.DAT. Valid RT-11 filenames consist of the optional device specification, a filename consisting of a maximum of six alphanumeric characters, and an optional filename extension of up to three alphanumeric characters. (For more detailed information see the RT-11 System User's Guide, pages 3-4..3-5).

(4.1) One final note on file manipulation deals with the user specified output files. At the completion of a normalization session, you will be asked to provide the filename where relations resulting form the session are to be stored. If only a partial list of functional dependencies has been created, (i.e. one or more relation has not been decomposed), these dependencies will automatically be saved by the tool in a separate file of the same name with a ".SVE" extension. For example, if you specify the output filename as OUTPUT.DAT, the existing relation information will be stored in OUTPUT.DAT, and any existing dependencies will be saved in file OUTPUT.SVE. At a subsequent normalization session, if you wish to resume work on file OUTPUT.DAT, the tool will search the appropriate drive for file OUTPUT.SVE, and if found, upload the dependency information into the system.

(5.0) <u>System Interface</u>

Control of normalization tool functions is achieved through the use of a menu driven, interactive system. A mouse is used for all menu and data item selection, with keyboard input being limited to entry of appropriate data filenames and for entry of the names of new relations resulting from the decomposition process.

After the tool is called up, you will first be asked to to enter the name of the input data file. Once the file is located and its data uploaded, you will then be asked to calibrate the mouse if it has not been previously initialized since system power up.

(5.1) <u>Mouse Calibration</u>. When prompted by the system, you calibrate the input device by placing the mouse in the upper right hand corner of its pad and sliding the mouse along the diagonal to the lower left hand pad corner. Hit any keyboard key when ready to continue. If the device is not properly calibrated, you will be instructed to repeat the process until accurate mouse-computer communication is established, and the mouse's origin is determined.

(5.2) <u>Mouse Operation</u>. If at any time during mouse operation, the mouse's position on its pad does not correspond to the screen's cursor position, or if the mouse is run off the edge of its pad, simply place the mouse in the upper right hand pad corner and slide it along the diagonal. Repeat this procedure until you observe the cursor drive to the lower left hand corner of the screen and the mouse and cursor postions are once again in agreement.

The mouse is used to select a specific relation to be normalized, individual attributes that comprise a functional dependency, and system menu items. When prompted by the system to make a selection, move the cursor to the desired item and depress the mouse center key. (The left and right keys have no function in the current implementation.) Any mouse selection will result in a visible change to information displayed on the screen, as well as new system directives displayed in the message area at the bottom of the primary CRT.

(5.3) Output Display. The normalization tool uses two CRT displays for the output of information. The primary (amber) screen displays all menu, relation and attribute information that you need to specify functional dependencies and to perform the non-loss decomposition of relations. The secondary (green) screen is used only for the display of requested on-line help, and to provide a composite view of existing (previously specified) functional dependencies.

The primary display is divided into four functional areas: the menu area, the system message area, the data display area, and the workspace area. The menu area contains the selectable menu items for a given tool level. The system message area, located at the bottom of the screen, gives current status information about the current menu item selected, provides simple prompts of expected user action, and displays applicable system error messages. The data display area contains listings of either relation names or attribute names depending upon the current tool level. A maximum of 15 names may be shown in the display area at any given time. If the input file contains more than 15 relations, or a selected relation has more than 15 attributes, the system provides an information paging capability, allowing access to all possible information. The workspace area is used primarily for the display of functional dependency diagrams in the editing phase of the tool. This region is also used for the display of system messages in other phases of operation.

(6.0) System Operation

Once the normalization tool is called up, the system has been designed to give you the information that you need to carry out the normalization process. Because the system is menu driven, you have flexibility to perform a variety of tasks. The information contained on the remaining pages is meant as a general guide to system use.

GUIDE TO SYSTEM OPERATION

(1). Turn the main (red) LSI-11 System "A" power switch ON.

(2). Ensure that the mouse power switch (AUX 1) is ON.

(3). Use the RT-11 system call ASSIGN to assign the system default drive to the device that contains program executible code and data files. (See paragraphs 3.0, 3.1).

(4). Type the command RUN FDTOOL.

(5). Enter the filename of the data file containing the relations to be normalized. (See paragraph 4.0)

(a). An error message indicates that the specified file can not be found on the default, or specified drive. Ensure that you have correctly entered the filename <u>and</u> its device name if the file is not on the default drive.

(b). If you are unable to enter a valid file exit to the operating system by typing X and locate the necessary files.

(6). Calibrate the mouse if directed to do so by the system. (See paragraph 5.1)

(7). Hit any mouse key to call up the main menu level.

(a). If no menu appears on the screen, the most likely cause is that the file MENU.TXT is not on the default drive. Exit to the operating system by typing CNTRL C and locate the menu file.

(8). Select the desired menu item.
(Because you have the option to select any of the following options at your discretion, the components of each menu level will be described, rather than giving a step by step procedure.)

(9). MAIN MENU Level:

(a). HELP Provides on-line help information for the menu items at the main tool level.

(b). EXIT TOOL Allows you to exit to the operating system at the end of the normalization session. This option may be selected at any time, with any work completed during the session being saved to user specified output files. (See paragraph 4.1) (c). NORMALIZE This is the primary tool option. After selection, you will be asked to choose the relation that you wish to normalize from the list of relations shown in the data display area. A relation name shown in reverse video has partial dependencies that have been specified during a previous session. An underlined relation name indicates that that relation has been previously normalized. A previously normalized relation may be selected assuming that real world constraints may have changed thus warranting further decomposition.

(d). PRINT RELATIONS This option allows you to obtain a hard copy listing of the input file's existing relations at the time the print option is invoked.

(10) NORMALIZE MENU Level:

(a). HELP Provides on-line help information for the NORMALIZE MENU options.

(b). EXIT Allows you to exit to the MAIN MENU level where you may either select a new relation to work with, get a print out of the existing relations or exit to the operating system.

(c). DECOMPOSE Allows you to execute the non-loss decomposition of the selected relation, based on the set of specified functional dependencies shown on the secondary CRT. Before selecting this option, ensure that you have specified all existing dependencies based on real world constraints. Also ensure that you have included all relation attributes in at least one of the specified dependencies. If you forget to include an attribute, the system will warn you that the attribute(s) is missing. This check is made to ensure that an attribute(s) is not lost from the relation set during the decomposition.

(d). EDIT FD This menu option allows you to specify new functional dependencies and/or modify existing dependencies shown on the secondary CRT.

(11).PRINT MENU LEVEL:

(a). HELP This option provides instructions for setting up the system printer to allow printing of a hard copy of the current relation set.

(b). EXIT This option allows you to return to the MAIN MENU Level, if you decide that you do not want to print the relations or if you are unable to access the system printer.

(c). START This option initiates the printing of a list of all relation names and relation attributes that exist at the time the print option is invoked. After the data is printed, the system automatically returns to the MAIN MENU Level.

(12).EDIT FD MENU Level:

The options at this level allow you to select and delete attributes that comprise a given functional dependency and to save or delete dependencies in the set of specified dependencies. All select or delete options are executed on the dependency diagram shown in the workspace area of the primary CRT. Thus to modify one of the dependencies shown on the secondary CRT, you must first bring that dependency into the workspace before editing. When the workspace area is empty, this indicates that there is no current dependency, and the system is ready for you to specify a new functional dependency as desired.

(a). HELP This option displays on-line help for EDIT FD MENU items.

(b). EXIT This option allows you to return to the NORMALIZE MENU Level. Any dependencies shown on the secondary CRT are saved by the system, and comprise the set of dependencies that the system will use if a subsequent decompostion is performed. Before selecting the EXIT option, ensure that you save the current dependency displayed in the workspace by using SAVE FD. A dependency that is not saved prior to exit will not be used if a decomposition is subsequently performed.

(c). DEL ATTRIBUTE This option is used to remove a specific attribute from the functional dependency diagram shown in the workspace. You will be asked to select the desired attribute from the data display area. You may continue to delete attribute names in this manner until you select another menu item.

(d). DEL FD This option allows you to delete the entire functional dependency diagram from the workspace. To ensure that you do not inadvertently delete the diagram, you will be asked to re-select the DEL FD option, at which time the deletion will be executed. If you do not wish to delete the diagram, simply select another menu option to continue.

(e). SEL FD This option is used to bring a previously specified functional dependency into the workspace from the secondary CRT. All existing dependencies displayed on the secondary CRT are referenced by a number. When prompted by the system, type the number of the dependency you wish to select. The selected dependency will be removed from the secondary CRT and re-displayed in the primary screen workspace. Once the dependency diagram is in the workspace, you may use any of the edit options to modify the diagram.

(f). SEL DETERMINANT This option is used to specify the determinant attributes for the dependency in the workspace. If the option is selected when there is no current diagram in the workspace, the system will generate a new functional dependency diagram with the selected attribute. Selection of determinant attributes may continue until another menu item is selected. If the option is selected after there is an existing diagram in the workspace, additional attributes may be added to the current list of determinant attributes.

(g). SEL DEPENDENT The features of this option are identical to those of SEL DETERMINANT, except that the selected attributes will be added to the list of dependent attributes for the current dependency.

(h). SAVE FD This option is selected when you want to save the dependency shown in the workspace, and add it to the list of existing dependencies. The workspace diagram will be erased, and the dependency re-displayed on the secondary CRT. When the workspace is clear, the system is ready to accept the input of another functional dependency through the use of the SEL DETERMINANT or SEL DEPENDENT options.

Appendix B

Evaluation of the Human-Computer Interface of a CAD Normalization Tool

The following questionaire is designed to provide feedback on the human-computer interface of the CAD normalization tool developed during my thesis effort. Through your responses, I hope to measure your degree of satisfaction with the design tool, with primary emphasis on the "user-friendliness" of the human-computer interface.

The questionaire consists of a list of 8 factors. I hope to obtain your reactions and attitudes, based on your response to six possible adjective pairs used to describe each factor. asked to indicate your feelings about the factor.

I would appreciate any specific comments that you would care to make about any of the factors or any additional comments about the CAD tool in general. Since you have been asked to use two differnt versions of the tool, please indicate which version this questionaire pertains to, and which version you used first.

THIS QUESTIONAIRE PERTAINS TO:
( ) Version 1 Non-menu driven.
( ) Version 2 Menu driven

I FIRST PERFORMED THE NORMALIZATION USING VERSION: 1 2

1. System Feedback: The extent to which the system kept you informed about what was going on in the program.

insufficient								sufficient
unclear								clear
useless								useful
bad								good
unsatisfactory								satisfactory
To me this factor is: unimportant								important

Comments:

2. Display of Information: The manner in which both program control and relation information was displayed to the user. Consider the use of two display screens.

confusing								clear
cluttered								well defined
incomplete								complete
complex								simple
unsatisfactory								satisfactory

To me this factor is:

unimportant								important

Comments:

3. Program Pacing: The speed with which you were allowed to perform the normalization task.

uncomfortable								comfortable
inconsistent								consistent
hindering								helpful
unreasonable								reasonable
unsatisfactory								satisfactory

To me this factor is:

unimportant								important

Comments:

4. Error Recovery: The extent and ease with which the system allowed you to recover from user induced errors.

unforgiving								forgiving
inconsistent								consistent
confusing								clear
difficult								easy
unsatisfactory								satisfactory

To me this factor is:

unimportant								important

Comments:

5. Data Input: The ease with which you were able to communicate with the CAD tool in terms of inputing data into the system. Consider calibration and manipulation of the input device.

uncomfortable								comfortable
hard								easy
inconvenient								convenient
inaccurate								accurate
unsatisfactory								satisfactory

To me this factor is:

unimportant								important

Comments:

6. Ease of Learning: Ease with which you were able to learn how to perform the normalization task using the CAD tool. Please consider any on or off line help that was available to you.

difficult								easy
confusing								clear
complex								simple
help unavailable								help available
unsatisfactory								satisfactory

To me this factor is:

unimportant								important

Comments:

7. Feeling of Control: Your ability to direct or control the activities performed by the normalization tool.

low								high
insufficient								sufficient
vague								precise
weak								strong
unsatisfactory								satisfactory

To me this factor is:

unimportant								important

Comments:

(8). System Usefulness : Your perception of how useful the system is as an aid to a data base administrator.

low								high
negative								positive
insufficient								sufficient
useless								useful
unsatisfactory								satisfactory

To me this factor is:

unimportant								important

Comments:

General Comments:

Thanks for your help !

Appendix C

Evaluation Handouts

LSI-11 SYSTEM INFORMATION

The following steps are to be accomplished to setup LSI-11 System "A", regardless of which version of the normalization tool you have been asked to use first. These steps will ensure that the system is configured for you to run either program NORM (menu driven) or P1NORM (non menu driven). All files that you will need for this exercise are located on the system hard disk drive, sector RK2:.

(1). The main (red) System "A" power switch should be on. The system will automatically boot and is ready for operation when you see the "." system prompt.

(2). The switch labeled AUX 1, adjacent to the main switch, should also be on. (This switch applies power to the mouse.)

(3). Type the system command: ASSIGN RK2: DK: This will assign the system default disk drive to hard sector RK2. This step will preclude you from having to specify device names in subsequent file manipulation.

(4). Typing the command DIR will display a directory of the exercise disk. You should see several files, including NORM.SAV, P1NORM.SAV and P2NORM.SAV.

(5). Version 1 (non menu driven) of the tool is called up by typing: RUN P1NORM. Type a carriage return at the ">" system prompt to begin the program. Ignore the system reference to the Votrax. It has been disconnected.

Version 2 (menu driven) of the tool is called up by typing: RUN FDTOOL

This will hopefully get you started on the right foot/disk!!! The attached sheets give you specific information that you will need for each version. User Guides for each version have also been placed at SYSTEM A if you care to use them. Please complete one of the attached questionaires after using each tool, and place them in the box adjacent to the system. Thanks for your help.

VERSION 2 DATA, USED WITH PROGRAM NORM (Menu Driven)

The relational information shown below is contained in file TEST.DAT and is to be used in conjunction with the menu driven program NORM. After calling up the normalization tool with the command RUN NORM, type the filename TEST.DAT when prompted by the system.

Two relations are contained in TEST.DAT; COMPANY_DATA and SALES_DATA. I would like you to use the CAD tool to normalize the SALES_DATA relation. To simplify your task, I am giving you the "real world" relationship or dependencies that exist among the attributes of relation SALES_DATA. Use the information shown below to specify functional dependencies and perform the nonloss decomposion of the original relation.

Semantic Assumptions:

* No two customers have the same shipping address.
* Each order is identified by a unique order number.
* Each line within an order is identifies by a line number that is unique to that order.

ORDER_NUMBER determines the DATE and ADDRESS.

ORDER_# and ORDER_LINE_# together determine the ITEM_#, the QUANTITY ORDERED and the QUANTITY_REQUIRED.

ITEM_# and PLANT_# together determine the QUANTITY_ON_HAND and the DANGER_STOCK_LEVEL.

ITEM_# determines ITEM_DESCRIPTION

ADDRESS determines CUSTOMER_#

CUSTOMER_# determines ANNUAL_SALES, CUST_BALANCE, CUST_CREDIT_LIMIT and CUST_DISCOUNT

VERSION 1 DATA USED WITH PROGRAM P1NORM (Non-Menu Driven)

The information shown below is to be used in conjunction with program P1NORM and P2NORM. This version of the tool automatically uploads relation information from a default file, so it will not be necessary to enter a file name. This program is divided into two parts: P1NORM is used to specify functional dependencies among relation attributes. Once you have specified dependencies, you then run program P2NORM which performs the non-loss decompostion of the original relation.

Shown below are the "real world" relationships or dependencies that exist among the attributes of the input relation. Use this information to specify the functional dependencies when running program P1NORM.

SNAME and LOCKER together determine TEAM_#.

LOCKER determines LOCKER_LOC.

TEAM_# determines PROJECT_GRADE and PROJECT_PHASE.

SNAME and LOCKER together determine COURSE_GRADE.

Appendix D

Summary Paper

Introduction

A computer database management system (DBMS) is a record keeping system whose purpose is to accurately record and maintain information (7:3). A database administrator (DBA) is responsible for the design, development and maintenance of such a system and has overall control of the stored information. Some of the DBA's design tasks are to define the contents of the database and to decide the schema or structure of the system. These tasks can be very complex and time consuming, particularly for a large database.

It may be possible to help the DBA perform some of these complex design tasks by developing computer aided design (CAD) tools. One such task is the normalization of relations, a process of establishing design constraints for a relational database system. In the normalization process, the DBA must define dependencies between the attributes of a database and combine the attributes into relations or tables. The application of normal forms (constraints) can help reduce data redundancies and leads to a high level of data reliability.

There are several factors that make the normalization process a reasonable candidate for the development of a design tool; the process is time consuming and prone to human error, as well as being fairly well defined and adaptable to automation. Travis (31) and Jankus (16) developed an interactive normalization software tool for use on the LSI-11 computer. This tool consisted of two main parts: an interactive package that enabled the user to specify functional dependencies from a given list of attributes, and a non-interactive portion that performed the actual normalization of relations once the functional dependencies were determined.

The interactive portion of the CAD tool used two CRT screens, one to display information and instructions to the user, the other to display a list of attributes from which the user determines functional dependencies. The tool provided the user with two selection techniques, using either the keyboard or a mouse, to select the determinant and dependent attributes for a given relation. The user could continue the selection process until all desired relations were specified.

The purpose of this effort was to enhance the CAD normalization tool with respect to both computer input and output. To solve this problem, it was first necessary to examine the manual process which a database administrator goes through in determining functional dependencies. It was then necessary to determine how a computer interface could be designed to better aid the DBA in accomplishing this task.

Approach

The approach taken in this study was to first evaluate the existing system in terms of user-friendliness. Jankus' CAD tool had been used by a group of students for a computer data base system class project. The critiques provided by these students were reviewed for feedback on the features of the tool which were helpful, and to determine those aspects of the tool which were either confusing or "unfriendly" from a user's point of view. With the student critiques in mind, a literature search was then conducted to gain a better understanding of current techniques used in the design of user friendly systems. Using the student feedback and insight gained from the literature review, modifications were then made to enhance Jankus' software, using the interface devices currently available in the AFIT Digital Engineering Laboratory. Finally, the modified CAD Normalization Tool was evaluated by a group of students, familiar with the normalization process, to provide feedback on the system modifications.

This paper will first discuss some of the problem areas of the existing normalization tool. Then, based on information found in current literature, a series of interface design guidelines will be presented. With an understanding of these guidelines, the software development process used in this effort will be described. Then, some of the project's more significant design issues will be presented. Finally, results of student evaluations of the re-designed normalization tool will be discussed.

Problem Areas

Prior to the re-design of the normalization tool interface, it was helpful to examine those aspects of the existing tool which were in need of improvement. No formal testing or evaluation phase had been carried out for the tool. However, a group of 30 Computer Systems graduate students, enrolled in a computer data base systems class, were assigned the task of using the tool to normalize a given relation. Comments provided by the students were very useful in identifying features of the CAD tool that were either confusing or not useful. In addition, the comments gave a strong indication of those aspects of the program that were something less than user-friendly. Interface features that were most frequently criticized can be categorized into six general areas. The following is a brief summary of each of these problem areas.

Usefulness. Nearly one-third of the students remarked that they found no advantage to using the tool over performing the task by hand. Several people commented that it was necessary to draw out the functional dependencies by hand before being able to enter the data using the CAD tool.

Consistency. Approximately one-third of the group commented that the program was inconsistent, primarily in the area of system input. Inconsistencies resulted in confusion and frustration for the user.

Error Tolerance. Six of the thirty students remarked that the tool was not very tolerant of errors.

On Line Help. The degree of program helpfulness (or lack thereof) was commented on by at least ten individuals. Remarks ranged from some of the instructions being confusing with not enough information, to complaints that there was no way to get shorter prompts for more experienced users.

System Output. Many students commented on the lack of useful output provided by the tool. Although the program successfully performed the normalization task, and formatted the output for use by other modules in the DBMS, output was not provided for the user in a format so that he could readily see and understand the product of the tool.

Hardware Distractions. Comments in this area pertained mainly to some of the devices used in the mouse prototype. Approximately half of the students complained that use of a speech synthesizer was annoying and added more confusion to the process than help. Likewise, several people commented on distraction caused by use of the mouse. The difficulties encountered by the individuals using the mouse prototype exemplify the problems that can arise if the available hardware devices are not carefully integrated into the human-computer interface.

Design Guidelines

The problem areas discussed above are not unique to software written in an academic environment. Rather, these critical areas, often slighted by even professional system designers, must be addressed in the design of user friendly software. In conducting a search of current literature, it became apparent that there is no cookbook approach that designers can use in developing friendly interactive systems. Designers must have an awareness of human factor issues, and if possible, use a set of design principles or guidelines that can help in the design of more friendly systems. Software designer Henry Simpson offers a series of twelve design guidelines which are representative of the diverse principles found in software literature (30:46-60). The following guidelines, as a minimum, must be considered by the designer:

(1). Define the user.

(2). Anticipate the environment in which the program is to be used.

(3). Give the operator control.

(4). Minimize the operators' work.

(5). Keep the program simple.

(6). Be consistent.

(7). Give adequate feedback.

(8). Do not over-stress working memory.

(9). Do not over-stress recall memory.

(10). Help the user remain oriented.

(11).Code information appropriately.

(12).Follow prevailing design conventions.

The application of these design principles can in no way guarantee the quality of the interface. However, more friendly systems may result if the designer can internalize these general guidelines and then apply them in specific design problems.

CAD Normalization Tool Development Process

In light of the design guidelines just presented, a complete redesign of the existing system was deemed necessary to attain a more user friendly CAD normalization tool. Although software modifications could have been made to add user friendly aspects to the existing code, such an approach would have been contrary to the precept that ease of use is a broad consequence of design rather than a specific feature that can be added to a system (27:9). Therefore a complete redesign was accomplished using Simpson's (30) eleven step program development process.

(1) Define the System Objectives

The objective of the CAD normalization tool is to provide the data base administrator with an interactive system through which he can specify functional dependencies that exist among attributes of a given relation. Using the specified dependencies, the tool should then perform a non-loss decomposition of the existing relation, normalizing it to third normal form.

(2) Define the System User

The designated user of this tool is a data base administrator, or a member of his staff. These individuals are computer professionals who have a thorough knowledge of the normalization process, and of the computer system on which the tool is to be implemented. The following assumptions were made concerning user characteristics, assumptions that influenced subsequent stages of the design process:

(a) As computer experts, the target users have a working knowledge of the system file structures and operating system commands.

(b) The system users will only be required to use the normalization tool on a periodic basis. That is, although they are familiar with the normalization process, the users may have extended periods of time between use of the tool.

(c) Speed and accuracy in performing the normalization task are of prime importance to the user. As the DBA regains familiarity with using the tool, he should not be burdened with excessive prompting or help features.

(d) Because the size of the relational files used by the DBA may be large, and due to the dynamic environment in which the tool will be used, the user can be expected to experience interruptions during the course of a normalization session. Therefore the system must enable the DBA to stop work at any point in the process and resume at a later time.

(3) Define the System Functions

Considering the previously defined system objectives and user characteristics, the normalization tool should provide the following functions or capabilities. The tool should:

(a) Allow the user to specify the name of an input file containing the relations to be normalized.

(b) Read the relational information from an input file previously created by DBMGR.

(c) Display the names of relations contained in a designated file, indicating their normalization status and the existence of any previously specified functional dependencies for each relation.

(d) Allow the DBA to select the relation that he wants to normalize.

(e) Display the list of attributes comprising the selected relation.

(f) Display a relation's functional dependencies that may have been specified during a previous normalization session.

(g) Allow the user to edit any existing functional dependencies, or to specify new dependencies.

(h) Give the user the capability to view any or all specified dependencies at any time during the editing process.

(i) Perform a non-loss decomposition of the current relation once the user indicates that all functional dependencies have been specified.

(j) Allow the user to choose a name for each new relation created as a result of the decomposition.

(k) Allow the user to quit at any time during the editing process, saving the current dependencies for use during a later session.

(l) Allow the user to specify the name of the output files used for storage of new relational data and any partially specified dependencies.

(m) Provide the capability to print a hard copy of all relational information at the user's request.

Given these basic system functions, a structured analysis and design technique (SADT) was used to graphically portray the upper design levels of the proposed system.

(4) <u>Plan the System Modules</u>

Continuing with the development process, the functions outlined in step three were used to derive system modules. A top down approach was used, taking the basic functions and breaking them down into logical pieces. Structure charts were used to specify the modular characteristics of the system design. At this stage of the development process, it became rather difficult to strictly adhere to the sequential steps of Simpson's development process. An orderly flow through each step of the process is certainly a goal to strive for, but is not always attainable due to the nature of the software development problem. For example, in designing the system modules, it was found necessary to at least consider the type of user interface that was desired for this application, and the type of hardware that would be used in the interface. A reasonable argument can be made that the next several steps of the development process will often occur simultaneously, rather than being treated as distinguishable steps in a sequential process. This situation is alluded to by Simpson when he states that his eleven step development process should not be regarded as "...a rigid formula that you must follow without variation. Rather, look upon it as an ideal. You must adapt it to fit your working style and the design problem that you are attempting to solve" (30:63).

(5) <u>Select the Hardware Configuration</u>

The AFIT Digital Engineering Laboratory System "A" , an LSI-11/73 running the RT-11 Version 5.1 operating system , was selected as the host computer for the CAD normalization tool. The primary reason for this selection was the fact that this system could support the use of a wider range of peripheral devices as part of the man-machine interface. Input devices considered for use in the normalization tool included a mouse, a joystick, and a graphics tablet. Other available hardware included a speech synthesizer and three display screens, one for normal CRT text display, one with high resolution graphics, and a third with medium resolution color graphics. Considering these devices, the decision was made to implement the mouse as a selection device in concert with the normal and monochrome graphics monitors for system output.

(6) Design the Human-Computer Interface.

The design of the human-computer interface involved consideration of the three aspects of program control, system output, and system input. The following discussion deals with each of these aspects as they relate to the normalization tool, and describes why the selected methods were chosen over others that were considered.

Considering the tasks to be performed by the tool, a menu driven method of program control was selected as being best suited for this application. In using the tool, the DBA performs the two relatively structured tasks of editing functional dependencies, and then executing the actual decomposition of an existing relation. In accomplishing these two tasks, there is a rather limited set of functions that must be performed, making a menu driven system feasible. A question and answer format or an operator initiated dialogue using a command syntax were also considered. However, because tool speed was considered to be important, and because the DBA was expected to use the tool only on a periodic basis, a menu format was considered to be best suited for the application.

The next task was to design how the menus and other information would be displayed to the user. Various screen designs were drawn by hand and evaluated to determine which provided the most desirable format for presenting information to the user. This design method, recommended by design experts of interactive systems, lets the system developer experiment with such factors as area shapes, area location and area boundaries before actually developing system code. In all prospective layouts, distinct functional information areas were planned, in an effort to provide consistency for the user, and to help the user remain oriented while using the system. The final screen design was coordinated with that of a second data base design tool being developed, to see if a common format could be used in the two CAD tools (10). It was felt that such commonality would add greater consistency to the various design components, and hopefully simplify the user's task in going from one tool to another.

The third interface design aspect, the way in which the user enters data into the computer, is provided by a mouse and through the use of the terminal keyboard. The mouse is the primary input device, used to make all menu, relation and attribute selections. Keyboard inputs are minimized to prevent unnecessary user transitions from the mouse to the keyboard.

(7) <u>Design the Data Structures and Files.</u>

To a large extent, the data structures and files used in this effort were pre-determined, due to the fact that code from a previous design was to be incorporated into the system. For example, the existing modules that perform the nonloss decomposition of relations use domain and relation data that is stored in a linked list structure of records. Each relation is comprised of linked structures that contain attribute information for a given relation. To ensure that new software was compatible with existing normalization tool code, and in the interest of developing modules that could be used in other applications which use the same domain and relation data, the data structures were not changed in this design effort.

Similarly, the formats used in input and output files used in this effort remained unchanged from those used in previous applications. This was necessary once again to ensure that the input and output data from the CAD normalization tool was compatible with other components of the data base management system.

(8) Prepare the System Specification

This phase of the development process involved reviewing the information developed in the previous seven steps, to ensure that the design of the system to this point was as complete as possible prior to beginning the coding phase. A formal specification was not drawn up. However, all design work was reviewed and modified where necessary to ensure that all system objectives were satisfied.

(9), (10), (11) Write, Document and Test Program Code.

The final three program development stages were accomplished concurrently, and are briefly discussed together. System coding was accomplished using the Whitesmith C programing language. A top down approach was used in coding the system modules. This method proved to be very compatible with the two-layered menu design of the system. For example, the main module, in addition to performing system setup and initialization, called subroutines which constituted the menu items of the system's main menu. These subroutines were developed and tested prior to moving to the next lower level. Likewise, the subroutines made calls to other routines that corresponded to menu items of layer two. Coding and testing modules in this manner continued through to the lowest module level of the system. Code documentation, in the form of AFIT/ENG standard module headers, was included in all program code as modules were written. Header information proved to be helpful during program development as coding changes were made, and serves as an aid for future system modifications.

Design Issues

The following discussion consolidates several design issues into three categories: CRT display, mouse implementation, and system file manipulation.

CRT Display Method

The first project design category that merits discussion is that of the techniques used for the display of information on the CRT. Information display was a very important issue in the design of a user-friendly system. The discussion that follows deals with some of the problems encountered in designing the CRT displays.

Two Heathkit H-29 Video Display Terminals were used in the normalization tool. The primary terminal was configured with a Cleveland Cadonics Graphics Card, providing graphics features not available on the basic H-29 (6). The card offered four different operational modes that could be used with the H-29 terminal: the H-29 standard terminal alphanumeric mode; the native graphics mode, used for generation of all graphics commands; the alphagraphics mode, used for entering text on graphic displays; and the tektronics mode which offered a wide range of graphics features.

The first version of code was created to generate the entire screen display, layout and text, in the terminal's graphics modes. The resulting display was not as clear as that generated in the standard terminal mode, and was characterized by a distracting visual flicker and a slower display of textual information. In addition, memory requirements for the graphics support software precluded pursuing this approach.

An alternative version was written to see if the functional dependency diagrams could be drawn strictly using the graphics set contained in the standard terminal mode. Such an approach was feasible because the set contained line segments that could be used to form the boxes and arrows required for the diagrams. Some flexibility would be lost in terms of drawing arrows at various angles; however this was not considered to be a serious drawback, because effective graphical representations could still be created using this pseudo graphics approach.

In using the approach, sufficient memory was saved so that it became possible to perform both the specification of functional dependencies and the non-loss decomposition of relations within one program, instead of separating the two routines as had been done in the Jankus prototype. Consolidating the two routines into one program contributed to making the tool easier for the DBA to use.

Mouse Implementation

A second major design issue raised during the project was that of developing software required to interface the mouse as the primary data input device. Jankus had used the mouse as an attribute selection device in his prototype. However, to use the mouse as a selection device for more than one type of data, i.e. menu items, relation names and attribute names, it was necessary to develop new modules capable of differentiating between the different types of data selected from the CRT.

The selection process was simplified by dividing the CRT into four functional areas, each area defined by specific line and column boundaries. The menu area, for example, occupies columns 64 thru 80 and lines 1 thru 21. Similarly, the message, display and workspace areas are bounded by specified column and line parameters within the system software. Although the display cursor appears to have free movement within all areas of the screen except the message area, modules FIND_SCREEN_LOCATION and MOWSE2SCREEN actually direct the cursor to one of four logical columns on the screen, columns 1, 21, 42 or 64 and line numbers that lie in the range of 1 to 21.

This four column scheme was used to simplify the task of selecting various items from the screen. For example, if the user has the option of either selecting a relation name from the display area or choosing a menu item, the fact that the selected column number is 64 indicates to the software that a menu choice has been made. Likewise if a returned column number is 1, 21, or 42, and the corresponding line number is between 12 and 21, the software recognizes that the choice must be a relation name. Having made this column determination, either module GET_MENUCHOICE or SEEK_RELATION is then called to determine the specific item selected within the functional area. In the case of a menu item, GET_MENUCHOICE uses the line number of the selected item to search a linked list data structure containing the names of the currently displayed menu, and identifies and returns the menu item selected. Module SEEK_RELATION uses a similar scheme to identify a specific relation name chosen from the display area.

System File Manipulation

A third design issue faced in the development of the normalization tool was the way in which the system manipulates data files. The original prototype restricted the user to a default file name that contained a set of relations to be normalized. In the re-designed system, the user is given the opportunity to choose the file that he wants to work on, giving him greater flexibility. Likewise, on exiting the tool, the user can specify the name of the file in which the current set of relations is to be stored, rather than overwriting the file that contained the original relations. Therefore, if the user wants to maintain the original set of relations that existed prior to the normalization session, he can do so simply by selecting a new name for the output file. Using this scheme, the DBA can set up his own archive system, keeping a record of all relations prior to any normalization.

The system also automatically creates files containing any partially specified dependencies created during a normalization session. If the user is working on a large relational set, and he wishes to exit and complete the normalization process at a later time, all previously specified dependencies for an unnormalized relation are automatically saved. The system appends a ".SVE" extension to the user-selected output filename, and stores the dependencies for future use. When the DBA later uses the tool and enters a filename, the system searches for that filename with a ".SVE" extension. If the file exists, the dependency data is uploaded for use in the current session. This file manipulation remains transparent to the user, so all the DBA has to worry about is the name of his desired input file, and the file in which he wishes to store his output information.

The normalization tool also uses file storage for all online help information displayed to the user. When the user selects a help menu item, the system software seeks and displays the appropriate help text file on the secondary CRT. In light of the memory limitations of the host system, this file scheme was selected so that the large amount of textual information would not have to be kept in internal memory. This method also allows changes to be made to the displayed help information, without having to search through the tool's source code. Similarly, all user prompts and system messages are maintained in a separate file, simplifying the task of making changes to the displayed information.

CAD Tool Evaluation

The purpose of this section is to describe the evaluation of the normalization tool by a group AFIT graduate students. First, the tool used to measure and analyze computer user satisfaction will be presented. Then, the methodology of the CAD tool evaluation will be described, followed by a description of the evaluation results.

Measurement Tool. There is very little information found in current literature to indicate the existence of proven tools to measure and analyze either user-friendliness, or system user satisfaction. Bailey and Pearson, however, present a technique to measure and analyze user satisfaction of information systems, a technique that would appear to be adaptable to other types of computer systems as well. Based on current behavioral science research, Bailey and Pearson contend that user satisfaction is the sum of one's positive and negative reactions to a set of factors relative to a given situation. Satisfaction can then be defined as the sum of the user's weighted reactions to a set of factors,

$$S = \sum_{j=1}^{n} R_{ij} \quad W_{ij}$$

where, R = the reaction to factor j by individual i, and W = the importance of factor j to individual i (22:531). Using this definition, it is necessary to determine the relevant factors (R), as well to develop a vehicle for scaling the individual's reaction to those factors.

Bailey and Pearson developed a list of 39 factors that various experts agreed were of concern to information system users. These factors covered a broad spectrum, ranging from the user's perception of his relationship with the EDP staff, to the user's feeling of confidence in the information system output (22:539-542). Using a semantic differential technique, four unique bipolar adjective pairs, separated by a seven interval scale, were devised to describe the characteristics of each of the 39 factors. In addition, two common adjective pairs were used for each of the factors: satisfactory - unsatisfactory, used to measure the internal consistency of the four unique adjective pairs; and important - unimportant, used to measure the weight given to the factor (W in the given equation).

The scale between each word pair was quantified by assigning the values -3, -2 .. +2, +3 to the intervals. The importance scale was assigned values from .10 to 1.00 in increments of 0.15. Using these values, the user's reaction to a given factor can be computed as the average response to the four adjective pairs:

$$R_{ij} = 1/4 \sum_{x=1}^{4} I_{ijk} \quad \text{where}$$

$$I_{ijk} = \text{the numeric response of user i to adjective pair k of factor j.}$$

Summing the individual weighted factor responses over all 39 factors, the overall user satisfaction becomes:

$$S_i = \sum_{j=1}^{39} W_{ij} \; / \; 4 \sum_{X=1}^{4} I_{ijk}$$

The range of S is from +117 to -117. Bailey and Pearson recommend a normalization of the resulting score to a range from -1 to +1. (See reference (21) for details.) The normalized scores can then be translated to yield a measure of user satisfaction.

Evaluation Methodology

The measurement and analysis techniques described in the previous section were adapted to gauge the degree of user satisfaction in the normalization tool developed in this project. Since the effort's main area of interest was the system's degree of user-friendliness, a questionaire was created, consisting of eight factors related to the human-computer. Using, a semantic differential technique, four adjective pairs were selected for each factor, along with the importance and satisfaction word-pairs used by Bailey and Pearson. The complete questionaire used in the evaluation is shown in Appendix B.

The majority of test subjects used in the tool evaluation were graduate students enrolled in AFIT EENG 793, Advanced Software Engineering. It was felt that the students' expertise in the areas of software design and development would be helpful in providing objective feedback on the interface of the normalization tool. The use of this group, however, did induce a degree of artificiality in the evaluation. The target user for the normalization tool was specified as an experienced DBA or a member of his staff, individuals well-versed in the normalization process. Although several of the test subjects were familiar with relational database concepts, most did not possess as in-depth a knowledge of the normalization process as would be expected of the target user. Therefore, some individuals experienced some minor problems in using the tool, problems stemming more from task unfamiliarity, than from CAD tool deficiencies.

Task unfamiliarity was anticipated when designing the test scenarios used in the tool evaluation. Prior to using the tool, all students were given a brief overview of the normalization process, and of the concept of functional dependencies. In addition, rather than requiring each student to determine the functional dependencies relevant to a given attribute set, each student was given the real world dependency information. (See Appendix C for sample evaluation materials made available to the students.) Therefore in using the tool, unlike a DBA who would draw upon his knowledge of the relation information to specify dependencies, the students were asked to simply use the tool to enter information that was already given them. Although somewhat artificial, it was felt that this aspect would not seriously impact the results of the evaluation. The main objective was to receive feedback on the user-friendliness of the software interface, feedback that was successfully obtained.

A total of sixteen students and two faculty members were asked to participate in the exercise, with fifteen individuals actually completing the evaluation questionaires. Each individual was asked to use two different versions of the CAD tool in normalizing a given relation; version one was the original tool and version two the re-designed system. Both versions were evaluated, not so much to determine which was the better of the two, but rather to gain an insight into the interface features that potential users found to be useful for the given application. Since each test subject was asked to use both versions, half of the group was asked to use version one first, and the other half was requested to first perform the task using version two. This was done in an effort to detect any possible influence that exposure to one system might have on the evaluation of the subsequent system.

After completing the normalization task using a given tool version, subjects were asked to complete a questionaire for that version before using and evaluating a second CAD tool. In addition to completing the formatted information on the questionaires, all individuals were encouraged to include any additional comments about the software interface.

Evaluation Results

Analysis of the evaluation data involved testing the validity of the measurement tool as well as analyzing the comparative results of the questionaire. The discussion that follows first addresses three categories of validity relevant to the measurement tool: content, predictive, and construct. Then, the comparative analysis performed on questionaire results will be described.

Questionaire Validation. The user responses to version two of the CAD tool were used to measure the validity of the questionaire. The first category examined was content validity. Content validity implies that all aspects of the attribute or factor being measured are considered by the tool (22:536). The attainment of content validity is to a large extent a subjective task that may depend on a trial and error process. Nunnally alludes to this fact when he says there are two major standards for ensuring content validity: a representative collection of items, and "sensible" methods of test construction. (21:92).

Pearson and Bailey offer a possible measure of content validity when they suggest a test of the correlation among the adjective pairs for each of the factors of the user-satisfaction questionaire (22:536). Correlation testing was performed on the four adjective pairs for each of the eight factors used in the this project's questionaire. The results of the testing were obtained by correlating the user responses for each adjective pair, with the sum of the responses for the other three word pairs for a given factor. (See Nunnally (21) for a more detailed description of the methodology). A T distribution was used to test the null hypothesis that the correlation values were positive. Only eight of the 32 possible adjective pairs had positve correlation values which were not significant at a .10 level, with the remaining pairs all showing significant correlation. These results would seem to indicate the trial and error nature of designing such a measuring device, and provide an indication of those elements of the questionaire that could be improved.

To gauge the predictive validity of the questionaire, each individual's response to a factor's unsatisfactory-satisfactory word pair was correlated with the computed satisfaction index for a given factor. Once again, a T distribution was used to test the hypothesis that positive correlation existed between the measured level of satisfaction and the respondent's self assessment of satisfaction. All eight correlation coefficients were found to be significant at a 0.01 level.

A third validation category, construct validity, relates to the extent that a variable of interest is abstract rather that concrete (21:96), and generally implies that a measuring instrument performs as expected, relative to the abstraction of the factor being measured (22:536). It was not clear what, if any, empirical measures could be used to guage the construct validity of the project questonaire, so this issue was not addressed in any detail. (For more information concerning this topic, see reference (21).)

Analysis. Having addressed the validity of the project's questionaire, attention turned to a comparative analysis of the questionaire's data. Initial observation indicated that version two of the CAD tool recieved a higher user-satisfaction rating than did version one. With the possible rating values ranging from -1.0 (maximally dissatisfied) to +1.0 (maximally satisfied), the results indicated that all respondents gave higher satisfaction ratings to version two than were given version one. Further analysis was conducted to test the statistical significance of the findings.

Analysis involved testing the null hypothesis, that there was no significant difference between the version one and version two results, versus the alternate hypothesis that version two received a higher rating of user-satisfaction than did version one. Due to the small sample size, a T Distribution was selected to test the hypothesis. In using the T Test, the assumption was made that the sample ratings for each version came from a normal distribution. A second assumption that should be made when using a simple T Test is that the individual observations from each sample are independent of one another. As mentioned earlier, the independence of the two ratings was questioned at the outset, because both versions were evaluated by the same group of people. Recognizing the possibility that the ratings of one tool could be biased by exposure to another, two different analysis techniques were used to reduce the impact of rating dependence.

The first technique used was a paired T test. Using this method, the measured variable was the difference between the two ratings for each of the fifteen test subjects. It was assumed that the differences (D-values) were normally distributed, and were independent. Using a .01 level of significance, with 14 degrees of freedom (d.f.), the T statistic used was 2.624. Since the computed T value was greater than the T statistic, the null hypothesis was rejected, lending support to the contention that the ratings given version two were higher than those given version one.

As an alternative analysis technique, the evaluation results were divided into two groups based on which tool a given test subject used first. Since ten individuals rated version 1 first, and five used version 2, the sample sizes used were reduced in size to 10 for version one and 5 for version two. The two samples were again assumed to be normally distributed and independent. An F test was used to verify the necessary assumption that the sample variances were equal. A two-sample T test was computed using an alpha value of .01 and 13 d.f., yielding a T statistic of 2.650 and a computed T value equal to 10.219. Since the computed T value was much greater than the T statistic, the null hypothesis was again rejected in favor of the alternate hypothesis that the level of user-satisfaction for version two was significantly higher than that for version one.

Conclusion

The purpose of this study was to re-design an existing computer aided design tool used to normalize the relations of a relational database. Primary emphasis was placed on researching state of the art techniques in human computer interfacing, and incorporating human factor concepts in the design of a more user-friendly system.

An appreciation was gained for the complexity of defining or describing what is meant by a user-friendly system. All too often, system users cannot even describe a friendly system until they see it, complicating the task of the system designer. Due to this fact, the software designer must have an awareness of certain design guidelines that can help him in the program design and development process.

Material discovered in the literature search, as well as practical experience gained in the design of the project's software, reenforced the fact that user-friendliness must be an integral part of the design process, and not something that can be added as an after-thought. The application of a few sound software engineering principles throughout the development process can go a long way in producing a more user-friendly system. The program development process presented in this report, although similar to other popular approaches, places greater emphasis up front on the design of the human-computer interface.

In developing the CAD tool software of this project, it was clear that certain design trade-offs may have to be made in creating a friendly interface. For example, the memory and graphics limitations encountered during the effort resulted in adjustments to the original interface design. The hardware limitations to some extent resulted in a re-thinking of the desired interface. Judging from material found in current literature, trade-offs of this type are not uncommon when designing interactive systems.

Finally, this project addressed the difficulties associated with trying to quantify the extent of user-friendliness, or in measuring the degree to which a user is satisfied with a computer system. The questionaire adapted to serve as a measurement tool in this project, should be viewed as a starting point from which other, more precise measuring devices may evolve.

Bibliography

1. Andriole, Stephen J. Interactive Computer-Based Systems. New York: Petrocelli Books, Inc., 1983.

2. Begg, Vivienne. Developing Expert CAD Systems. London: Kogan Page, 1984.

3. Card, Stuart K. The Psychology of Human-Computer Interaction. Hillsdale, N.J.: Lawrence Erlbaum Associates, Inc., 1983.

4. Carey, Tom. "User Differences in Interface Design," Computer 12: 12-20 (May 1982).

5. Carlson, Eric D. and others. Display Generation and Management System for Interactive Business Applications. Braunschweig, Weisbaden: Friedr. Viewy & Sohn, 1981.

6. Cleveland Cadonics User's Manual. Cleveland Cadonics Inc., 1982.

7. Date, C.J., An Introduction to Database Systems. (Third Edition) Massachusetts: Addison-Wesley Publishing Company, 1982.

8. Dean, Morris. "How a Computer Should Talk to People," IBM Systems Journal, 21: 424-453 (April 1982).

9. Dudley, Timothy K. "Computer and Graphics: A Technology of Age, Part II," Interactive Computer Graphics Systems, edited by William C. House. New York: Petrocelli Books Inc., 1982.

10. Finch, Richard. A CAD Tool to Aid the Database Designer in the Conceptual Phase of Database Design. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1985.

11. Foley, J.D.,and V.L.Wallace. "The Art of Actual Graphic Man-Machine Conversation," Proceedings of IEEE 62: 462-471 (April 1974).

12. Giloi, Wolfgang K. Interactive Computer Graphics, Data Structures, Algorithms, Languages. Engelwood Cliffs, N.J.: Prentice Hall, 1978.

13. Heathkit Manual for the Video Display Terminal Model H-29. Heath Company, 1983.

14. Heckel, Paul. The Elements of Friendly Software Design. New York: Warner Books, Inc., 1984.

15. Heines, Jesse M. Screen Design Strategies For Computer-Assisted Instruction. USA: Digital Equipment Corporation, 1984.

16. Jankus, Edward R. Development of Computer Aided Database Design and Maintenance Tools. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1984.

17. Kearns, Timothy. Implementation and Analysis of a Microcomputer Based Relational Database System. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1984.

18. Miller, George A. "The Magical Number Seven Plus or Minus Two: Some Limits on Our Capability for Proccesing Information," The Psychological Reveiw, 63: 81-97 (March 1956).

19. Moran, Thomas P. "The Psychology of Human-Computer Interaction," ACM Computing Survey, 13: (March 1981).

20. Norman, Donald A. "Design Rules Based on Analyses of Human Error," Communication of the ACM, 26: 254-258 (April 1983).

21. Nunnally, Jum C. Psychometric Theory. New York: McGraw Hill Book Company, 1978.

22. Pearson, Sammy W, and J.E. Bailey. "Development of a Tool for Measuring and Analyzing Computer User Satisfaction," Management Science, 29: 530-544 (May 1983).

23. Peters, Lawrence J. Software Design: Methods and Techniques. New York: Yourdon Press, 1981.

24. Raduchel, William J. "A Professional's Perspective on User Friendliness," Byte, 9: (May 1984).

25. Raeder, Georg. "A Survey of Current Graphical Programming Techniques," Computer, 18: 11-25 (August 1985).

26. Roman, David R. "Building Up Your Personal Computers," Computer Decisions, 110-128 (March 1984).

27. RT-11 Software Support Manual, AA-H379B-TC Operating System RT-11 Version 5.1, Digital Equipment Corporation, Maynard, Ma, March 1983.

28. Rubenstein, Richard, and Harry Hersh. The Human Factor, Designing Computer Systems for People. USA: Digital Equipment Corporation, 1984.

29. Shneiderman, Ben. "The Future of Interactive Systems and the Emergence of Direct Manipulation," Human Factors and Interactive Computer Systems, edited by Hannis Vassiliou. Norwood, N.J.: Ablex Publishing Corporation, 1984.

30. Simpson, Henry. Design of User Friendly Programs for Small Computers. New York: McGraw Hill Book Company, 1985.

31. Travis, Charles. Interactive Automated Systems for Normalization of Relations. MS Thesis. School of Engineering, Air Force Institue of Technology (AU), Wright-Patterson AFB OH, 1983.

32. Ullman, Jeffrey D. Principles of Database Systems. Rockville, MD: Computer Science Press, 1982.

33. Woffinden, Duard Stephen. Interactive Environment For A Computer Aided Design System. MS Thesis. Naval Post-Graduate School, Monterey, CA, June 1984.

VITA

Captain Thomas C. Mallary was born on 25 January 1953 in Clinton, Iowa. He graduated from Clinton's St. Mary High School in 1971 and completed one year of college at St. Thomas College in St. Paul, Minnesota. He received an appointment to the United States Air Force Academy in 1972, graduating in June of 1976 with a Bachelor of Science Degree in International Affairs and Business Management.

Captain Mallary completed Under Graduate Navigator Training and Electronic Warfare Officer Training in September 1977. He was then assigned to the 27th Tactical Fighter Wing, Cannon AFB, New Mexico, serving as an F-111D Weapons Systems Officer and Wing Electronic Warfare Officer until January 1981. He served as an Instructor Weapons Systems Officer with the 48th Tactical Fighter Wing, RAF Lakenheath, England, until his assignment at the School of Engineering, Air Force Institute of Technology in May of 1984.

Permanent address: 568 First Avenue
Clinton, Iowa 52732

AD-A164100

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/ENG/85D-8	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering — 6b. OFFICE SYMBOL *(If applicable)* AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS *(City, State and ZIP Code)* Wright Patterson AFB, OH 45433	7b. ADDRESS *(City, State and ZIP Code)*
8a. NAME OF FUNDING/SPONSORING ORGANIZATION — 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS *(City, State and ZIP Code)*	10. SOURCE OF FUNDING NOS.
11. TITLE *(Include Security Classification)* See Item 19	

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

12. PERSONAL AUTHOR(S)
Thomas C. Mallary, B.S., Capt USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Yr., Mo., Day)*	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	1985, December	161

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR.
09	02	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Database, Computer Aided Design, Relational Database, Human-Computer Interface

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

11. TITLE: DESIGN OF THE HUMAN COMPUTER INTERFACE FOR A COMPUTER AIDED DESIGN TOOL FOR THE NORMALIZATION OF RELATIONS

Approved for public release: IAW AFR 190-1/.
[illegible] 16 JAN 86
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED [✓] SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER *(Include Area Code)*	22c. OFFICE SYMBOL
Dr. Thomas C. Hartrum	513-255-3576	AFIT/ENG

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

The project involved the design and implementation of the human-computer interface of a computer aided design (CAD) tool used in conjunction with a relational database. The tool gives a database administrator an interactive means of specifying functional dependencies for a given relation. It then executes a nonloss decomposition, normalizing the original relation to third normal form.

Background information is provided on the fundamentals of a relational database and on the concept of functional dependencies. An understanding of the term "user-friendly system" is developed, together with a summary of state-of-the art techniques and guidelines meant to help the system designer create more friendly interfaces.

The program development process used in the project is described in detail. Major design issues of the project are described, with emphasis placed on areas impacting the system's human-computer interface.

A tool is developed to measure and quantify the extent of user satisfaction in the software system. The measurement tool is implemented by a test group asked to evaluate the normalization tool interface. Results of the evaluation are presented.